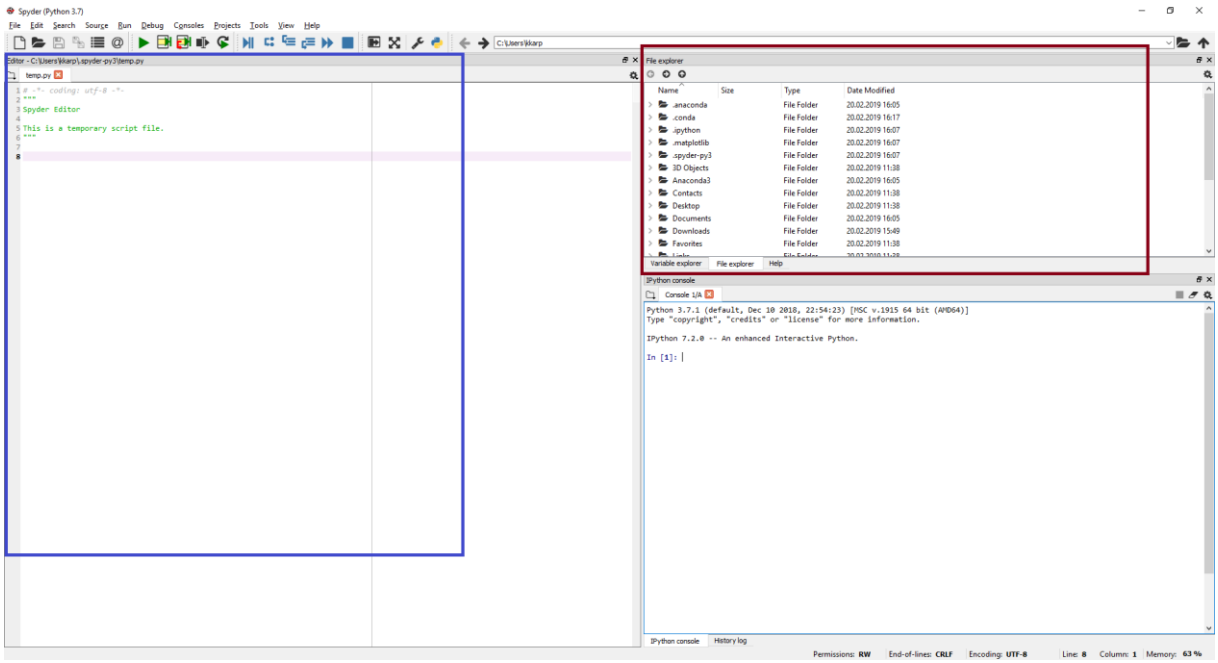
Laboratorium 4 – Regresja, część III

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 (Position_Salaries.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 wektorów wspierających (*support vector regression – SVR*)**

   Wstęp.

   Ten rodzaj regresji jest wykorzystywany w algorytmie maszyny wektorów wspierających (*support vector machines – SVM*). Jest to algorytm, który uczy się rozpoznawać określone zestawy danych przynależące do określonego atrybutu wyjściowego.

   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.

Zadania.

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

2.1. Wczytać zbiór danych Position_Salaries.csv.

2.2. Utworzyć macierz zmiennych niezależnych (Level) oraz wektor zmiennych zależnych (Salary).

2.3. Dokonać zamiany jednowymiarowej tablicy y na dwuwymiarową. Jest to wymóg, który należy spełnić, żeby w kolejnym kroku utworzyć obiekty sc_X i sc_y.

```
y=y.reshape(-1, 1)
```

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

```
from sklearn.preprocessing import StandardScaler

# Utworzenie obiektów sc_X i sc_y.

sc_X = StandardScaler()
sc_y = StandardScaler()

# Dopasowanie za pomocą metody fit_transform

X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

2.5. Dopasować model SVR do zestawu danych wykorzystując klasę SVR.

```
from sklearn.svm import SVR

# Tworzenie obiektu klasy SVR z argumentem kernel='rbf', czyli
ze wskazaniem że funkcja jądra 'rbf'

regressor=SVR(kernel='rbf')

# Dopasowanie modelu

regressor.fit(X, y)
```

2.6. Wykreślić utworzony model.

```
plt.scatter(X, y, color = 'red')
plt.plot(X, regressor.predict(X), color = 'blue')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

# Wykreślenie danych z większą rozdzielczością

X_grid = np.arange(min(X), max(X), 0.01
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
```

```
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

2.7. Wykorzystać utworzony model do predykcji parametru Salary dla Level = 6.

```
y_pred =
sc_y.inverse_transform(regressor.predict(sc_X.transform(np.array([[6.0]]))))
```

2.8. Wykonać sprawozdanie z realizacji podpunktów 2.1 – 2.7. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:

- Wykres z podpunktu 2.6
- Wartość obliczoną w punkcie 2.7

3. Drzewo decyzyjne (*Decision tree*)

Wstęp.

Drzewo decyzyjne tworzy modele regresji lub klasyfikacji w postaci struktury drzewa. Dzieli zestaw danych na mniejsze i coraz mniejsze podzbiory, a jednocześnie powiązane drzewo decyzyjne jest stopniowo rozwijane. Ostatecznym wynikiem jest drzewo z węzłami decyzyjnymi i węzłami liści.

Zadania.

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

3.1. Wczytać zbiór danych Position_Salaries.csv.

3.2. Utworzyć macierz zmiennych niezależnych (Level) oraz wektor zmiennych zależnych (Salary).

3.3. Dopasować model drzewa decyzyjnego do zestawu danych wykorzystując klasę DecisionTreeRegressor.

```
from sklearn.tree import DecisionTreeRegressor
# Tworzenie obiektu klasy DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
# Dopasowanie modelu
regressor.fit(X, y)
```

3.4. Wykreślić utworzony model.

```
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

3.5. Wykorzystać utworzony model do predykcji parametru Salary dla Level = 6.

```
arr=np.array([6])
arr=arr.reshape(1,-1)
y_pred = regressor.predict(arr)
```

3.6. Wykonać sprawozdanie z realizacji podpunktów 3.1 – 3.5. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:

- Wykres z podpunktu 3.4
- Wartość obliczoną w punkcie 3.5

4. Random forest

Wstęp.

Algorytm random forest umożliwia wykonywanie zadań regresji i klasyfikacji z wykorzystaniem wielu drzew decyzyjnych oraz techniki tzw. pakowania. Pakowanie tą metodą polega na szkoleniu każdego drzewa decyzyjnego na innej próbce danych. Wynik otrzymuje się po uśrednieniu rezultatów z każdego z drzew.

Zadania.

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

4.1. Wczytać zbiór danych Position_Salaries.csv.

4.2. Utworzyć macierz zmiennych niezależnych (Level) oraz wektor zmiennych zależnych (Salary).

4.3. Dopasować model random forest do zestawu danych wykorzystując klasę RandomForestRegressor.

```
from sklearn.ensemble import RandomForestRegressor
# Tworzenie obiektu klasy RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
# Dopasowanie modelu

regressor.fit(X, y)
```

4.4. Wykreślić utworzony model.

```
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

4.5. Wykorzystać utworzony model do predykcji parametru Salary dla Level = 6.

```
arr=np.array([6])
arr=arr.reshape(1,-1)
y_pred = regressor.predict(arr)
```

4.6. Wykonać sprawozdanie z realizacji podpunktów 3.1 – 3.5. Przesłać w formie pliku pdf do serwisu moodle. Plik powinien zawierać następujące informacje:

- Wykres z podpunktu 4.4
- Wartość obliczoną w punkcie 4.5