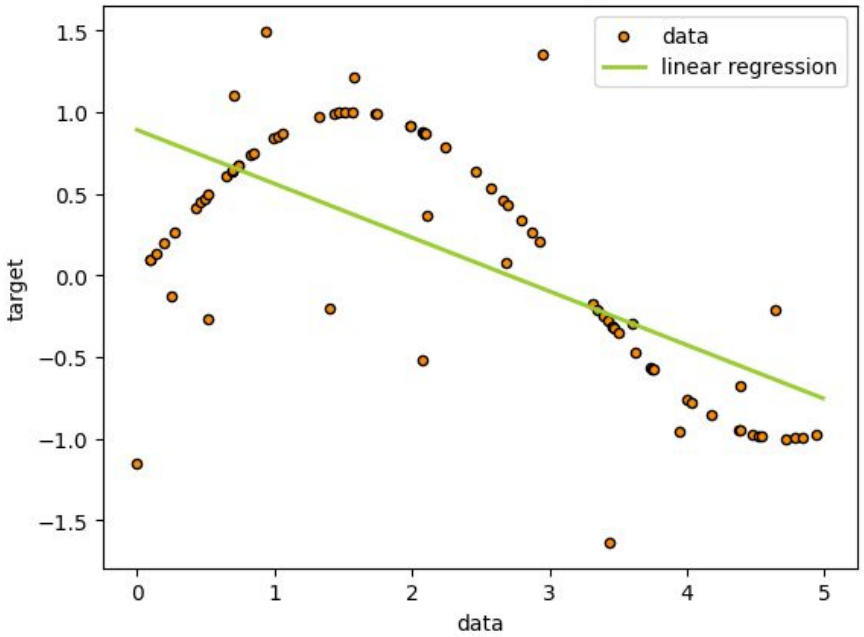
Drzewa decyzyjne - wstęp

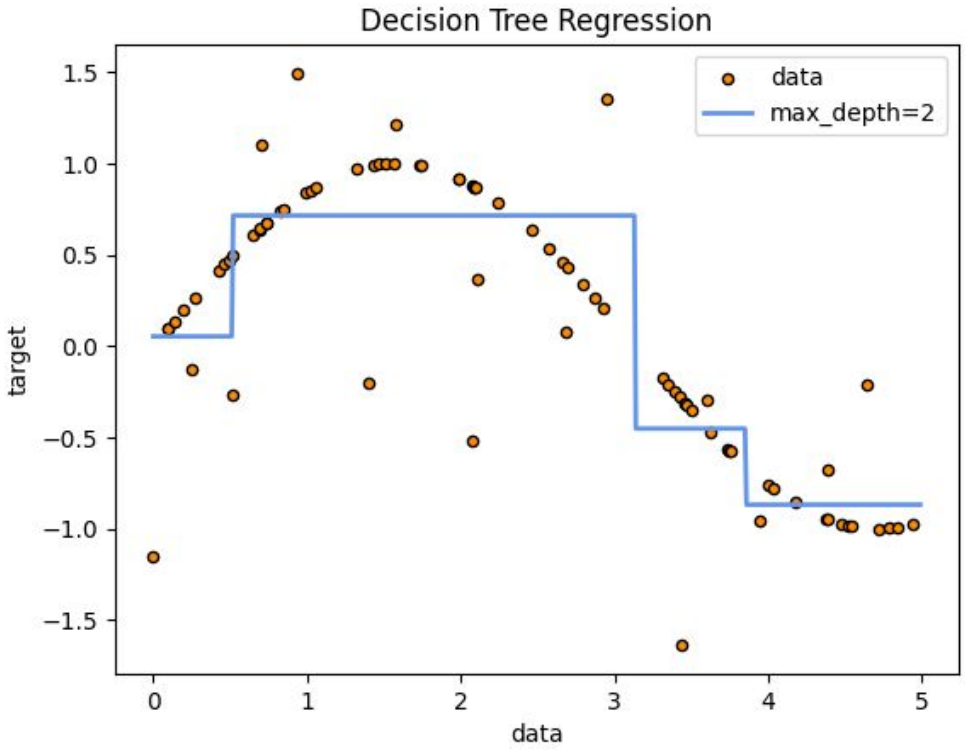
Marcin Wierzbiński

Regresja liniowa



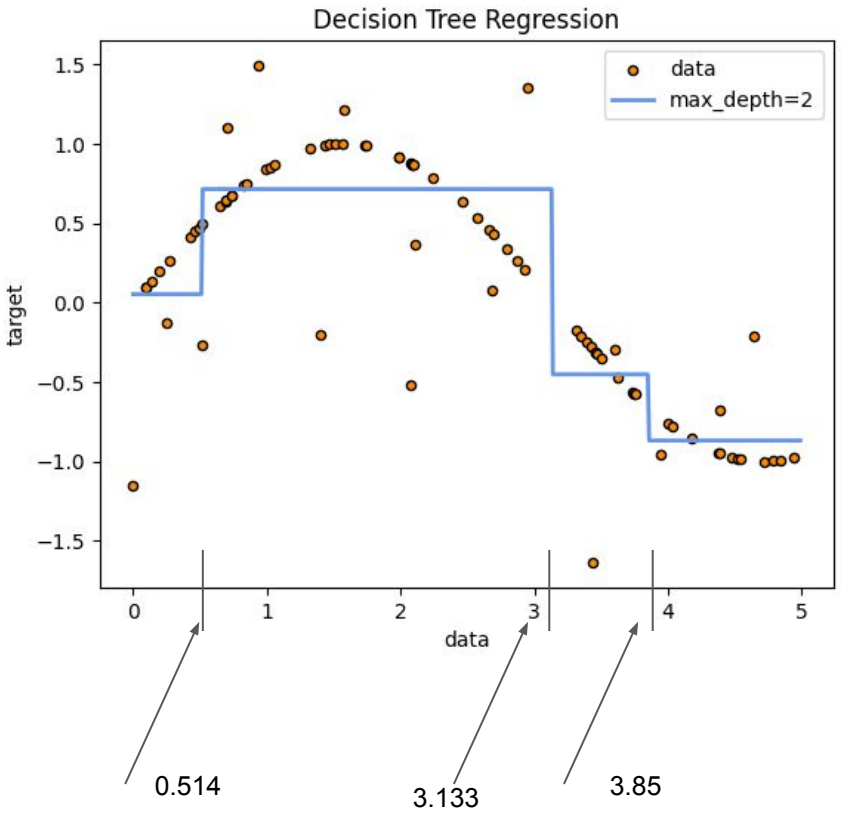
- Regresja liniowa zakłada liniową zależność między zmiennymi objaśniającymi a zmienną celu.
- Gdy nie ma takiej liniowej zależności, model regresji liniowej może być mniej skuteczny lub nieefektywny w przewidywaniu wartości zmiennej celu.
- Istnieją także algorytmy regresji nieliniowej, które pozwalają na modelowanie zależności między zmiennymi objaśniającymi a zmienną celu w sposób nieliniowy.

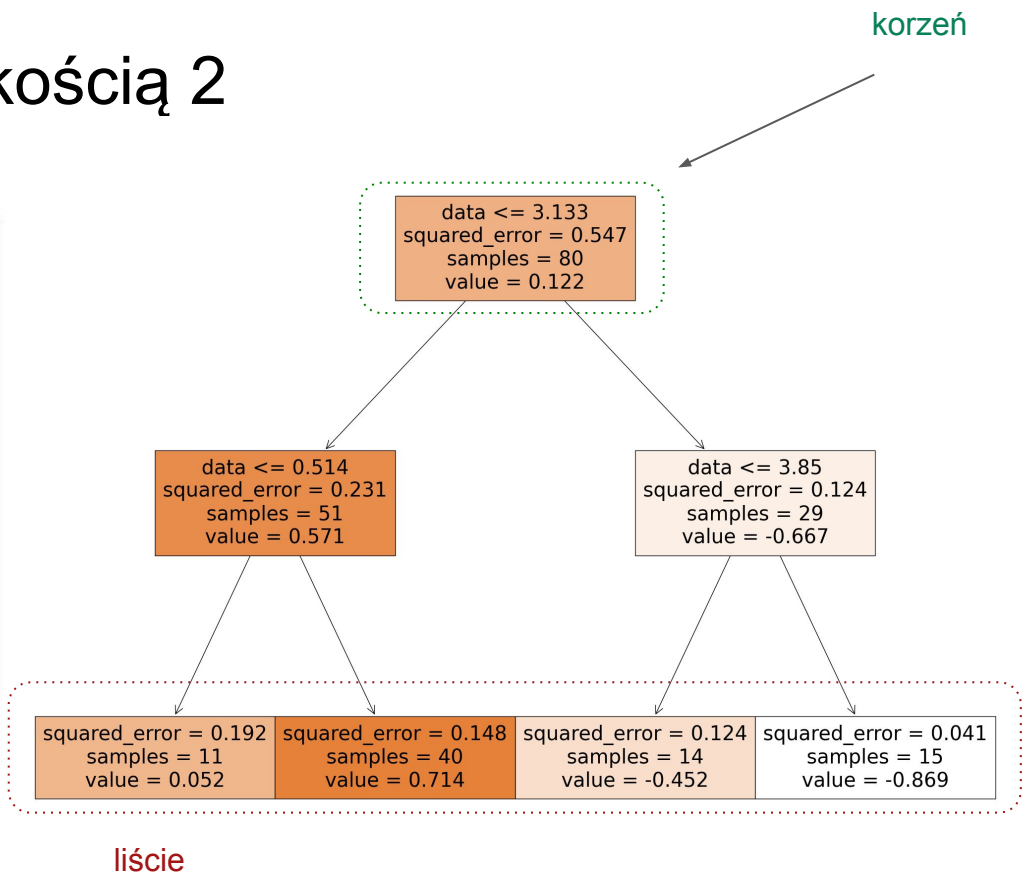
Decision Tree Regression



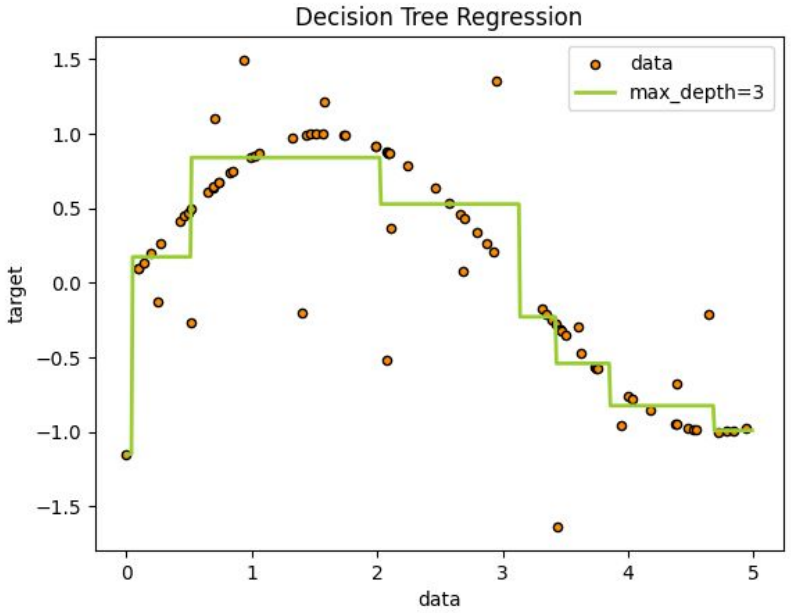
- Pozwala na modelowanie zależności między zmiennymi objaśniającymi a zmienną celu w sposób nieliniowy.
- DecisionTreeRegression buduje drzewo decyzyjne, które dzieli zbiór danych na coraz mniejsze podzbiory w zależności od wartości wybranej zmiennej.
- W przypadku algorytmu DecisionTreeRegression, podział następuje tak, aby zminimalizować wariancję w każdym podzbiorze.

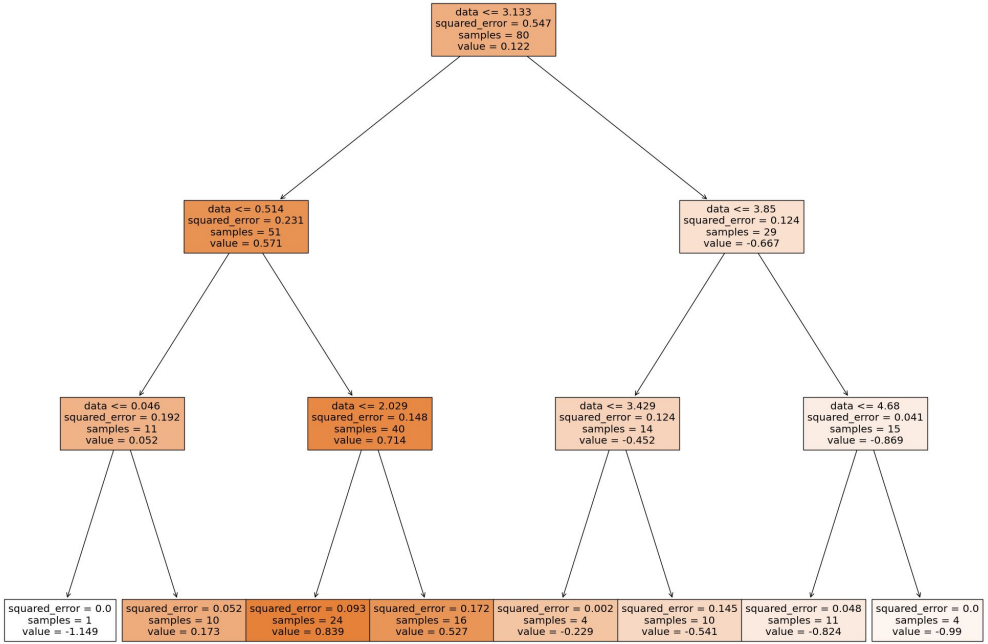
Drzewo decyzyjne z głębokością 2





Drzewo decyzyjne z głębokością 3





Implementacja Sklearn

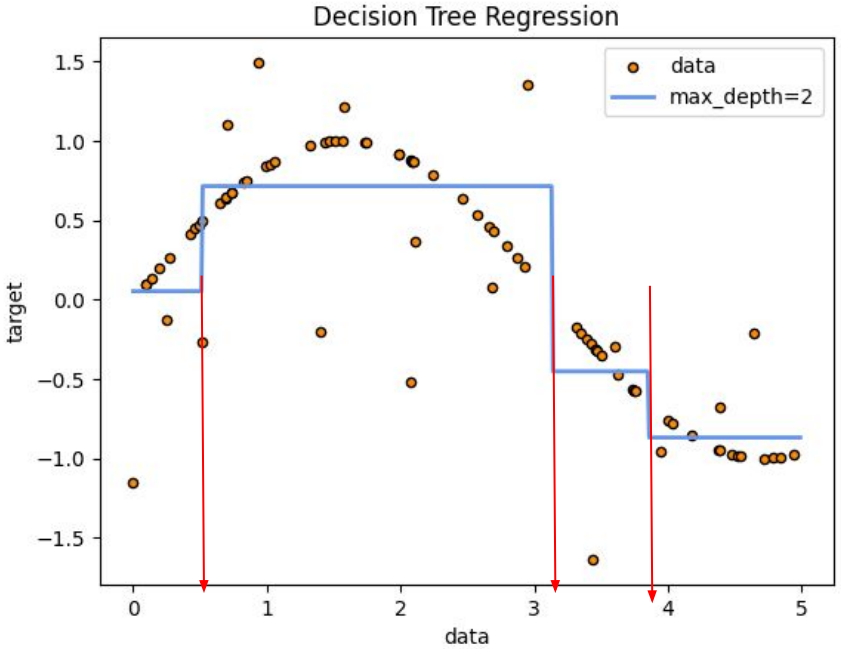
```
# podział na zbiór treningowy i testowy
X = data[['total_bill', 'tip', 'size', 'smoker', 'day', 'time']]
y = data['sex']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# tworzenie i trenowanie modelu
model = DecisionTreeClassifier(max_depth=4, random_state=42)
model.fit(X_train, y_train)
accuracy_train = model.score(X_train, y_train)

# predykcja na zbiorze testowym i obliczenie dokładności
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Dokładność testowa: {:.2f}%".format(accuracy*100))
print("Dokładność treingowa: {:.2f}%".format(accuracy_train*100))
```

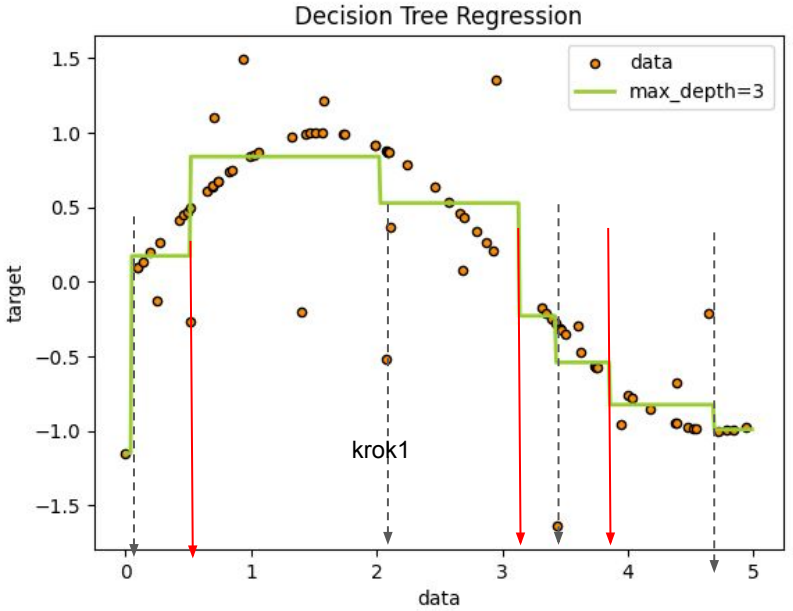
Algorytm Decision Tree Regression

1. Wybiera zmienną, która najlepiej dzieli zbiór danych na podzbiory, korzystając z miary jakości podziału, takiej jak MSE (Mean Squared Error) lub MAE (Mean Absolute Error).
2. Tworzy węzeł drzewa, reprezentujący tę zmienną i wartość progową.
3. Dzieli zbiór danych na dwa podzbiory, jedno zawierające obserwacje, których wartość dla wybranej zmiennej jest mniejsza lub równa wartości progowej, a drugie zawierające obserwacje, których wartość jest większa niż wartość progowa.
4. Rekurencyjnie powtarza ten proces dla każdego nowo utworzonego podzbioru, aż do osiągnięcia jednego z warunków zakończenia, takiego jak maksymalna głębokość drzewa, minimalna liczba obserwacji w liściu lub brak istotności statystycznej w kolejnych podziałach.

Po zbudowaniu drzewa, algorytm DecisionTreeRegression może być wykorzystany do prognozowania wartości numerycznych dla nowych obserwacji, poruszając się po drzewie od korzenia do liścia i przypisując wartość numeryczną liściowi, do którego dotrze.

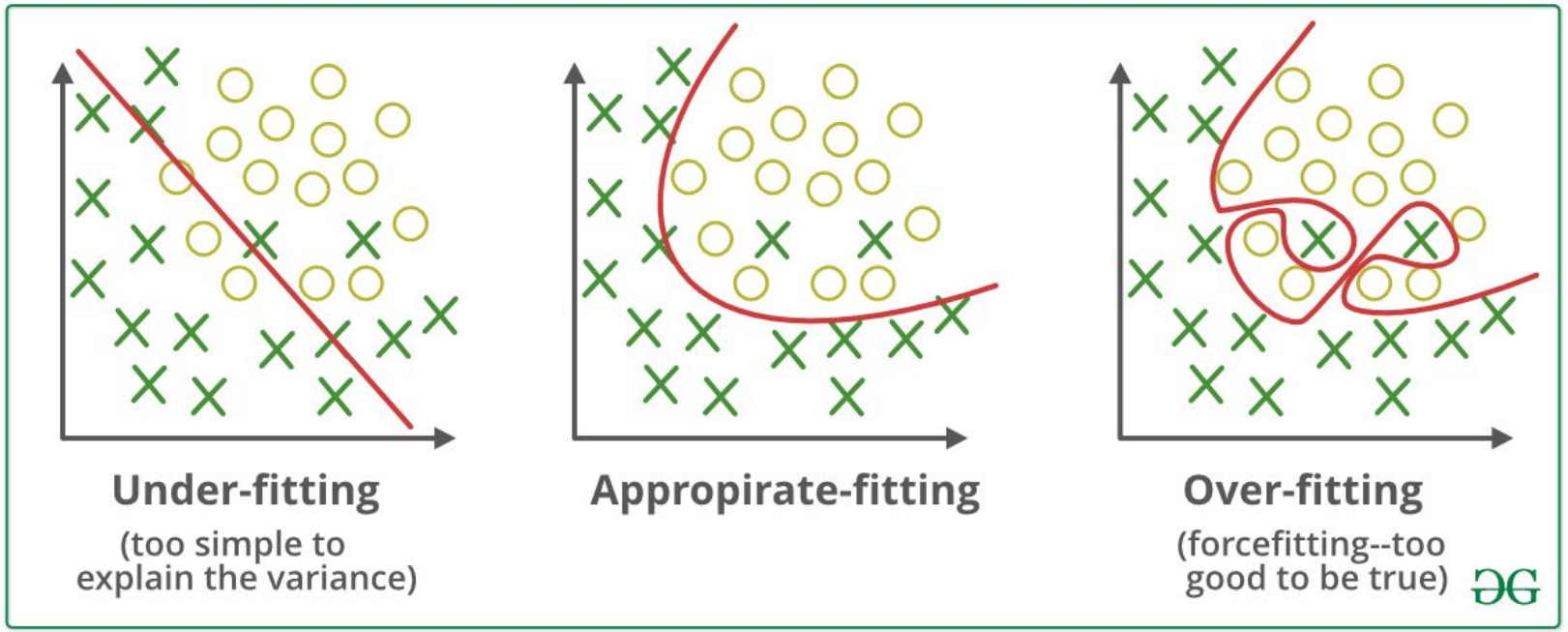
Przykład algorytmu:



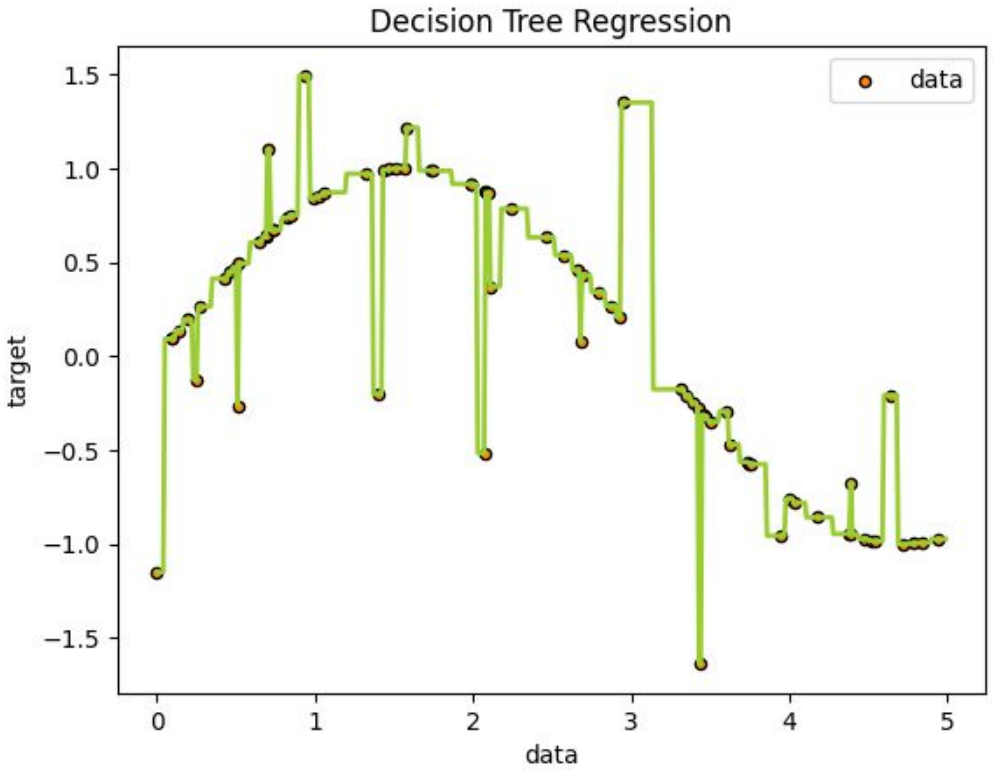


Rekurencyjny podział na 2 podzbiory

Overfitting problem

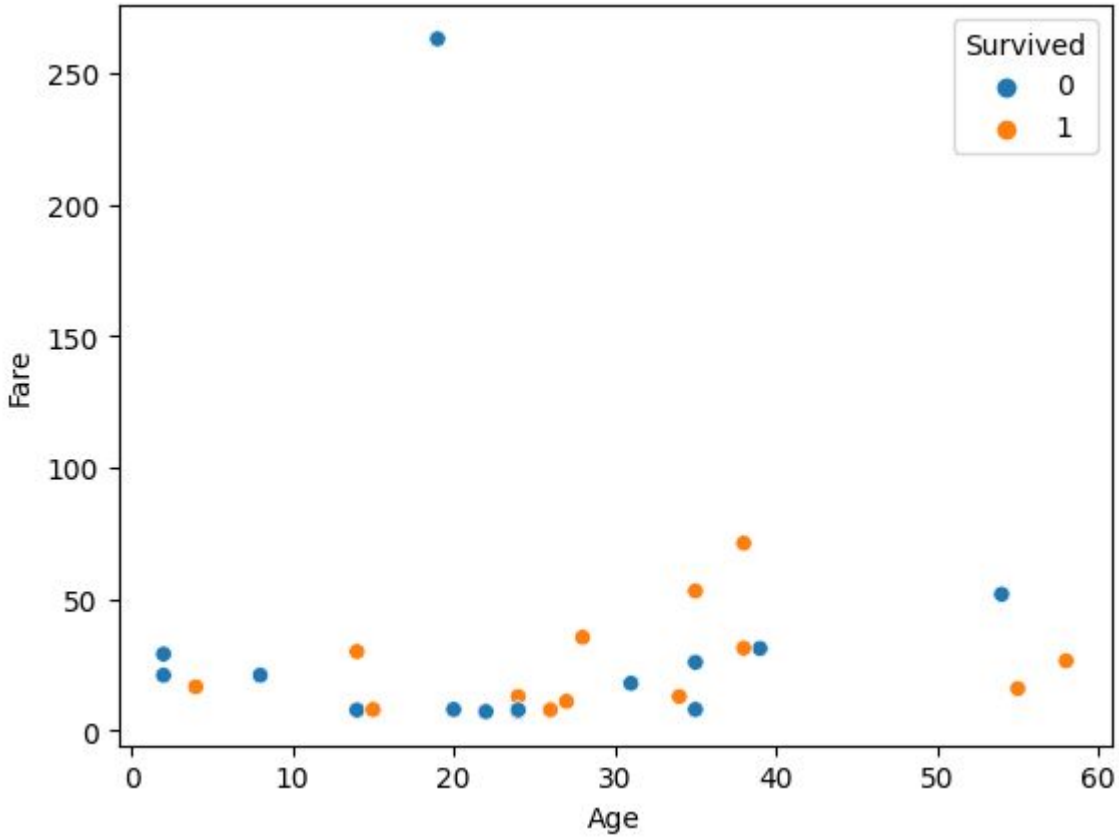


Decision Tree Regression overfitting




```
Dokładność treningowa: 1.00
Dokładność testowa: 0.71
max_depth = 10
```

Problem klasyfikacji



Funkcja loss:

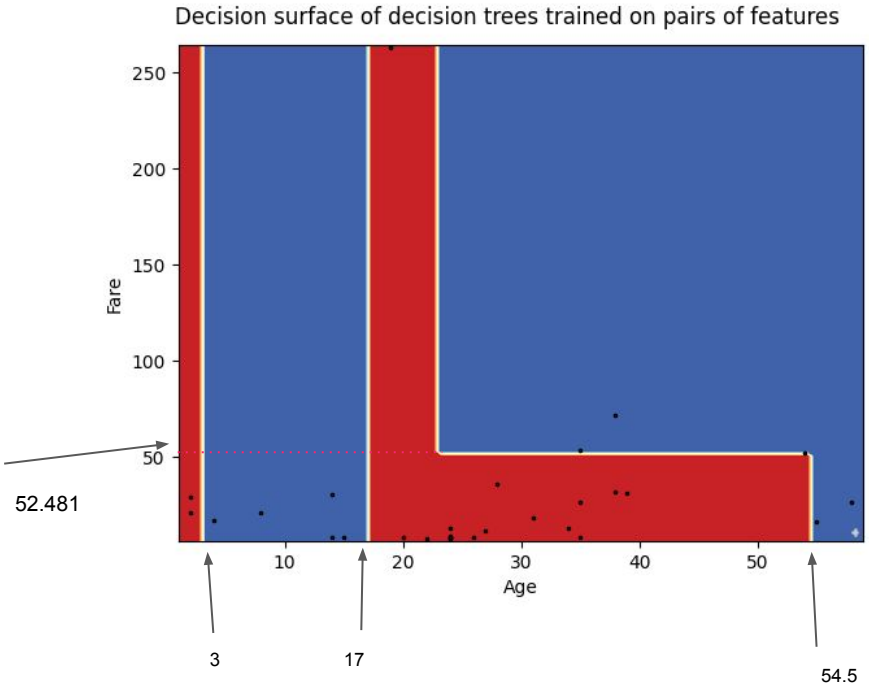
Regresja:

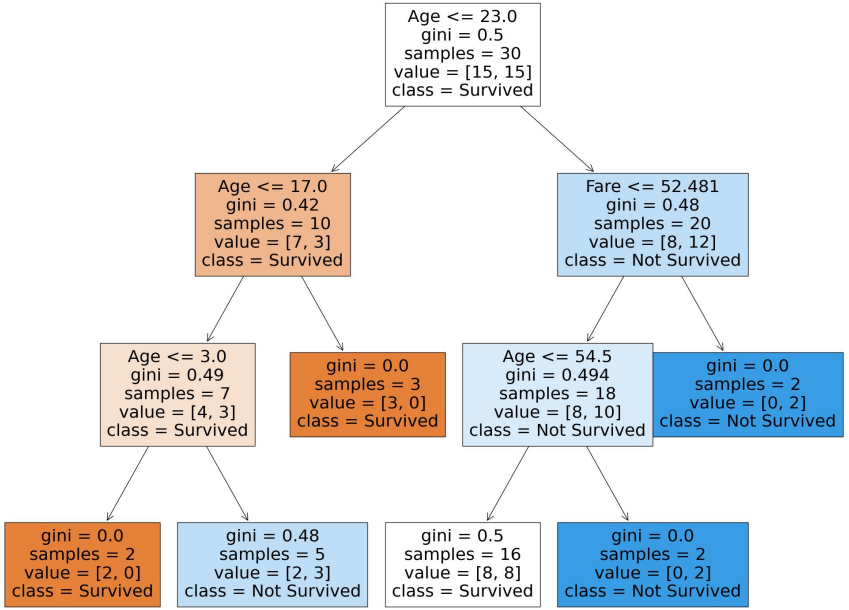
Błąd średniokwadratowy

Klasyfikacja

Entropia skośna

Algorytm Decision Tree Classification





predict: Survived, Not Survived

RandomForest

Algorytm Random Forest (losowy las) jest metodą zespołową (ensemble) uczenia maszynowego, która polega na łączeniu wielu drzew decyzyjnych w celu zwiększenia dokładności i redukcji wariancji modelu.

Algorytm Random Forest działa w następujący sposób:

- Losowo wybiera próbkę danych z dostępnej próbki treningowej.
- Z losowo wybranej próbki tworzy drzewo decyzyjne.
- Krok 1 i 2 powtarza się kilka razy (domyślnie 100 razy), aby uzyskać wiele różnych drzew decyzyjnych.
- W przypadku problemów klasyfikacji, każde drzewo w lasie decyzyjnym dokonuje predykcji dla danej próbki i głosowanie większościowe decyduje, która klasa zostanie przypisana do danej próbki. W przypadku problemów regresji, wyniki predykcji każdego drzewa są uśrednione.

Kluczową cechą algorytmu Random Forest jest fakt, że każde drzewo w lesie jest tworzone na podstawie losowego podzbioru danych treningowych.

Random Forest Algorithm

```
# tworzenie i trenowanie modelu
model = RandomForestClassifier(n_estimators=3, random_state=42, max_depth=3)
model.fit(X_train, y_train)

# predykcja na zbiorze testowym i obliczenie metryk
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
accuracy_train = model.score(X_train, y_train)
```

Modyfikacja tips

```
smoker = {'Yes':1, 'No': 0}
day = {'Fri':1, 'Sat': 2, 'Sun': 3, 'Thur': 0}
time = {'Dinner': 0, 'Lunch': 1}

import numpy as np

data['smoker'] = data['smoker'].map(smoker)
data['day'] = data['day'].map(day)
data['time'] = data['time'].map(time)
```

Modyfikacja titanic

```
data = pd.read_csv('titanic.csv', sep=',')

sex = {'male':1, 'female': 0}
embarked = {'S': 0, 'C': 1, 'Q': 2}

data['Embarked'] = data['Embarked'].fillna(np.random.choice(['S', 'C', 'Q']))
data['Age'] = data['Age'].fillna(np.random.randint(18, 35))
data['Sex'] = data['Sex'].map(sex)
data['Embarked'] = data['Embarked'].map(embarked)
data.drop('Cabin')
```

References

- https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html