Podstawy Sztucznej Inteligencji (PSI)

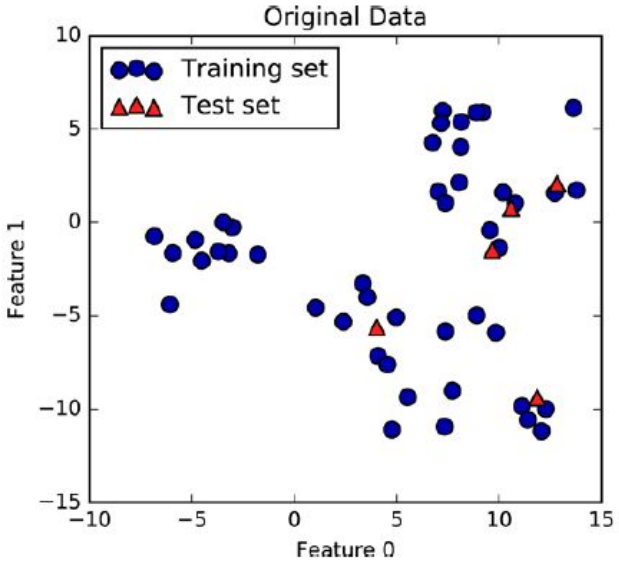
Lab 1 - omówienie

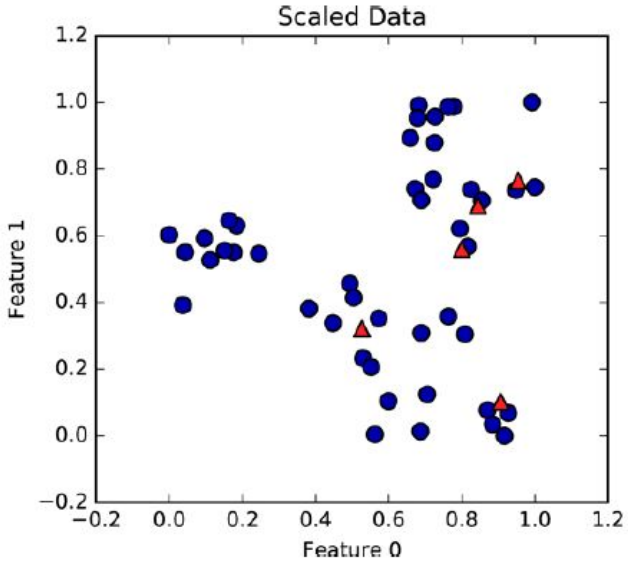
Pytania? Wątpliwości? Uwagi?

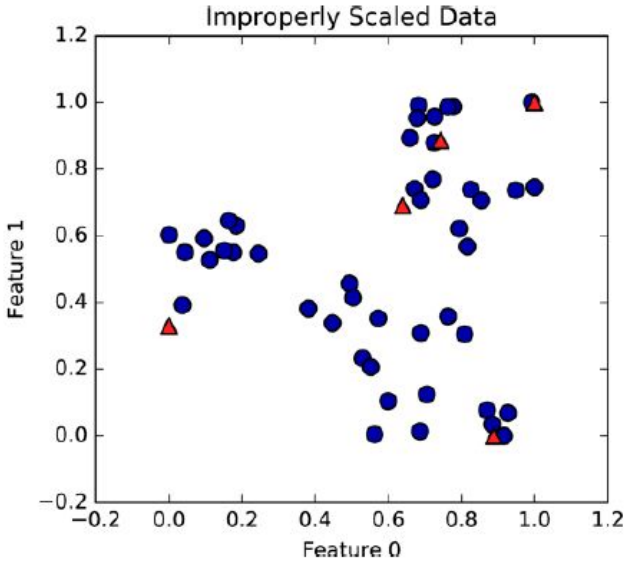
Częste problemy

Skalowanie danych testowych

- **trzeba** używać tego samego scalera na danych treningowych i testowych!







Skalowanie danych testowych

```
# WRONG - don't to this!
train_scaler = MinMaxScaler()
X_train = train_scaler.fit_transform(X_train)
test_scaler = MinMaxScaler()
X_test = test_scaler.fit_transform(X_test)

# PROPER - this way!
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

inplace=True

- nie ma sensu używać, po prostu
- https://stackoverflow.com/questions/43893457/understanding-inplace-true-in-pandas/59242208#59242208

Importy

```
import missingno as msno
```

- warto sprawdzać kod bibliotek, tutoriale, Githuba
- zwykle biblioteki mają konwencję importowania ich - trzymajmy się tego

Chaining warunków w Pandasie

- przetwarzanie danych to typowo proces sekwencyjnego "rzeźbienia" DataFrame'a
- najlepiej robić to funkcyjnym pipeline'm

```
rows_to_drop = train_data[
    (train_data.Embarked_C == 0) &
    (train_data.Embarked_Q == 0) &
    (train_data.Embarked_S == 0)
].index
train_data = train_data.drop(rows_to_drop)
```

Usuwanie listy kolumn

- można usuwać wiele naraz
- jest szybciej, szczególnie gdy musimy usunąć wiele kolumn - unika robienia wielu kopii

```
train_data = train_data.drop(columns=["Name", "Ticket"])
```

Formatowanie kodu

- np. warto trzymać się jednego tworzenia stringów, czyli cudzysłowów
- warto używać formattera kodu, który ujednolici formatowanie za nas
- black - wiodący formatter PSF, używa go np. Django

```
pip install black[jupyter]

black .
```

Pandas i wartości brakujące

- wszystkie standardowe metody statystyczne (mean, median, stdev etc.) domyślnie ignorują wartości brakujące
- `skipna=True`

Nazewnictwo zmiennych

- skalary i wektory małą literą

```
y_train
```

```
x_test_sample
```

- macierze i tensory dużą literą

```
X_train
```

```
X_test
```

Metryki w klasyfikacji

Celność (accuracy)

- podstawowa miara jakości
- wyznacza, jak często model przewiduje prawdziwą klasę
- **problem:** nie działa dobrze, kiedy liczność klas jest bardzo różna - **klasyfikacja niezbalansowana (imbalanced classification)**

$$Accuracy = \frac{TP + TN}{P + N}$$

	Actual 1	Actual 0
Predicted 1	True Positive (TP)	False Positive (FP)
Predicted 0	False Negative (FN)	True Negative (TN)

Precyzja (precision)

- jak wiele z pozytywnych predykcji jest faktycznie pozytywnych?
- jak bardzo możemy ufać modelowi, kiedy przewiduje on klasę 1
- **use case:** detekcja spamu, FP dużo gorsze niż FN

$$Precision = \frac{TP}{TP + FP}$$

	Actual 1	Actual 0
Predicted 1	True Positive (TP)	False Positive (FP)
Predicted 0	False Negative (FN)	True Negative (TN)

Czułość (recall / sensitivity)

- jak dużo faktycznie pozytywnych wartości udało się wykryć?
- jak bardzo możemy ufać, że model wykrył wszystkie przykłady z klasy pozytywnej
- **use case:** diagnoza w medycynie, FN dużo gorszy niż FP

$$Recall = \frac{TP}{P} = \frac{TP}{TP + FN}$$

	Actual 1	Actual 0
Predicted 1	True Positive (TP)	False Positive (FP)
Predicted 0	False Negative (FN)	True Negative (TN)

F1 score

- średnia harmoniczna precision i recall
- agreguje te metryki w jedną dla łatwiejszego porównywania
- **zalety:** wygodna, bardzo penalizuje ekstremalnie niskie wyniki (średnia harmoniczna)
- **wady:** nie bierze pod uwagę TN

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

	Actual 1	Actual 0
Predicted 1	True Positive (TP)	False Positive (FP)
Predicted 0	False Negative (FN)	True Negative (TN)

ROC i AUROC

- https://mlu-explain.github.io/roc-auc/
- nazwy: ROC AUC, AUC, AUROC
- "The AUC is the probability that the model will rank a randomly chosen positive example more highly than a randomly chosen negative example."
- **ROC use case:** analiza klasyfikatora, dobór progu
- **AUROC use case:** klasyfikacja niezbalansowana, bo skupia się na klasie pozytywnej (mniejszościowej)

Klasyfikatory drzewiaste

Drzewa decyzyjne

- https://mlu-explain.github.io/decision-tree/
- zbiór nauczonych if-else'ów
- obecnie ekstremalnie rzadko nie używane pojedynczo, zamiast tego tworzy się z nich **klasyfikatory zbiorowe (ensembles)**
- ekstremalnie niski bias i wysoka wariancja, zawsze wymaga regularyzacji

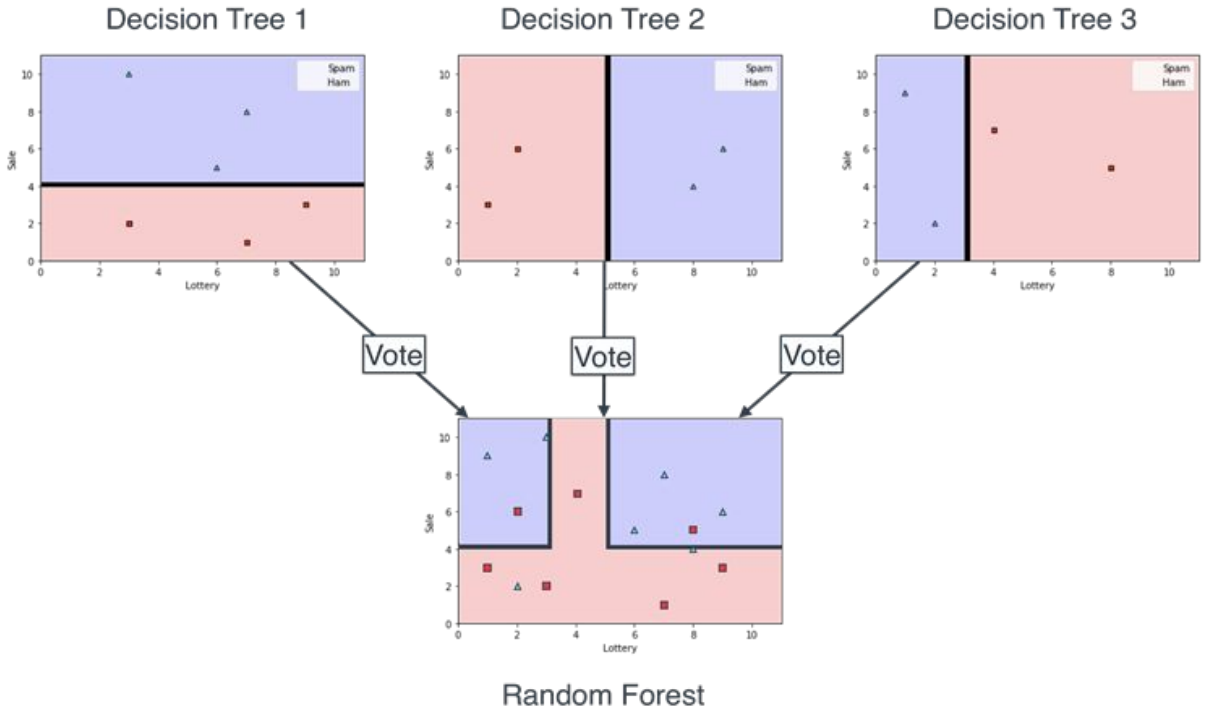
Drzewa decyzyjne - zalety i wady

- **zalety:**
  - prostota, interpretowalność
  - szybkość, skalowalność
  - działa dla dowolnych rodzajów zmiennych, w tym dla wartości brakujących (ale uwaga na implementację!)
- **wady:**
  - przeciętne wyniki
  - silnie preferuje (biased) atrybuty o dużej liczbie wartości (numeryczne)

Lasy losowe (Random Forests)

- https://mlu-explain.github.io/random-forest/
- wiele drzew decyzyjnych trenowanych **niezależnie** na **próbkach boostrapowych (bootstrap samples)**
- wynik przez głosowanie
- **zmniejsza wariancję** - po prostu, nie zmienia biasu
- nie przeucza przy zwiększaniu liczby drzew - to tylko "mocniej" uśrednia
- świetny **klasyfikator domyślny**

Random Forest - granica decyzyjna



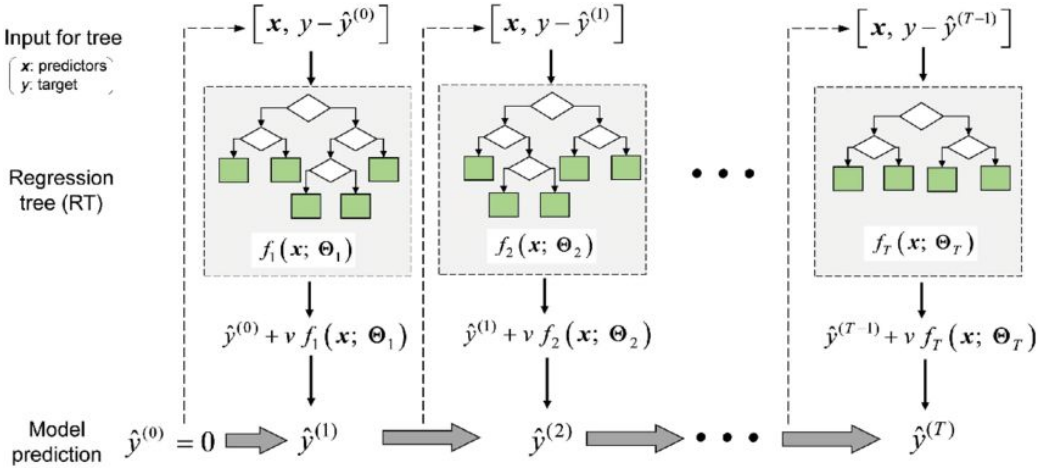
Random Forest - zalety i wady

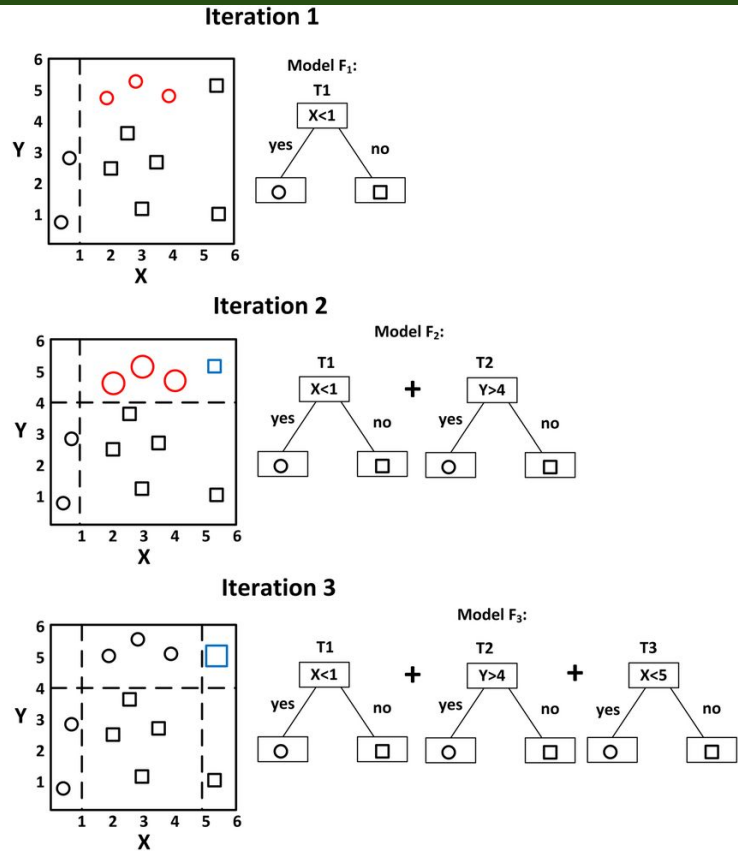
- **zalety:**
  - bardzo dobre wyniki przy domyślnych hiperparametrach
  - skalowalność (embarassingly parallel)
  - niska czułość na dobór hiperparametrów
  - prostota
  - działa dla dowolnych rodzajów zmiennych, w tym dla wartości brakujących (ale uwaga na implementację!)
- **wada:** nie wykorzystuje wektoryzacji procesora

Boosting

- wiele drzew decyzyjnych trenowanych **sekwencyjnie** na tym samym zbiorze
- kolejne uczą się na błędach poprzedników, więc **minimalizujemy funkcję kosztu** - dodajemy kolejne drzewo tak, żeby zminimalizować błąd popełniany przez cały ensemble
- spadek wzdłuż gradientu - **gradient boosting**
- wynik przez głosowanie
- **zmniejsza wariancję oraz bias** - dzięki dostosowywaniu kolejnych drzew
- **łatwo przeucza**, dużo hiperparametrów, głównie związanych z regularyzacją

Boosting - wizualizacja





Boosting - zalety i wady

- **zalety:**
  - często najlepsze wyniki
  - szybkość
  - skalowalność
  - działa dla wartości brakujących i zmiennych kategorycznych (ale uwaga na implementację!)
- **wady:**
  - łatwo przeucza
  - duża czułość na dobór hiperparametrów
  - dużo hiperparametrów