Strojové učení s využitím regrese a klasifikace

Strojové učení
Strojové účení je podoblastí oboru umělé inteligence, který se zabývá vývojem algoritmů schopných učit se z dat. Občas není možné pro daný problém vytvořit algoritmus, předpovědi, rozpoznání obličejů nebo věcí atd… Na rozdíl od tradičního programování, kde explicitně definujeme pravidla, u strojového učení necháváme samotný počítač, aby si pravidla odvodil sám na základě příkladů (data - správně i špatně)

Princip
Hlavním principem strojvého učení je hledání funkce, která mapuje vstupy a výstupy. Počítač na základě trénovacích dat snaží nalézt takovou funkci, která dobře funguje na trénovacích datech a zároveň dokáže správně předpovídat i pro nová data (generilazce)

Každý algoritmus pro strojové učení má využití a každý může být lepší na něco jiného. K nejvýznamnějším algoritmům patří Lineární regrese, Neuronová síť, Rozhodovací stromy atd…

Typy strojového učení

1. Supervised Learning

Pracuje s označenými daty, kde je znám správný výstup. Algoritmus se učí mapovat vstupy na známe výstupy. Dělí se na:
Klasifikace - zařazení do kategorií (rozpoznání objektu na obrázku)
Regrese - predikce číselné hodnoty (předpověď ceny nemovitosti)
Lineární regrese, rozhodovací stromy, neuronové sítě

2. Unsupervised Learning

Pracuje s neoznačenými daty, není definován správný výstup. Algoritmus sám hledá skryté vzory, struktury v datech. Detekce anomálií, Klastrování, Redukce

3. Reinforcement Learing

Neučí se z existujících dat, ale učí se z interakcí. Algoritmus provádí akce v prostředí a získává odměny/tresty. Učí se strategii, která maximalizuje celkovou odměnu. Robotika, autonomní vozidla, hraní her

Proces strojového učení

Základní proces zahrnuje:

1. Sběr trénovacích a testovacích dat
2. Příprava dat – čištění, transformace
3. Výběr modelu – volba vhodného algoritmu
4. Trénování modelu – učení z trénovacích dat
5. Evaluace – ověření kvality modelu na testovacích datech
6. Nasazení a monitoring – použití modelu v praxi

Regrese
Jedná se o metodu strojového s učitelem (Supervised learning) u které je cílem predikovat číselnou hodnotu na základě dat. Výstupem regrese je číselná hodnota - predikce.

Druhy regrese
Lineární regrese
Lineární regrese je metoda, díky které můžeme najít přímku tak, aby všechna data k ní byla co nejblíže. Lineární regrese používá lineární kombinaci

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta\square x\square + \varepsilon$

y = výstup funkce
x1,x2.. = nezávislé proměnné (prediktory)
β_0 ,β_1… = vliv jednotlivých proměn (koeficienty)
ε = náhodná chyba (residuální složka)

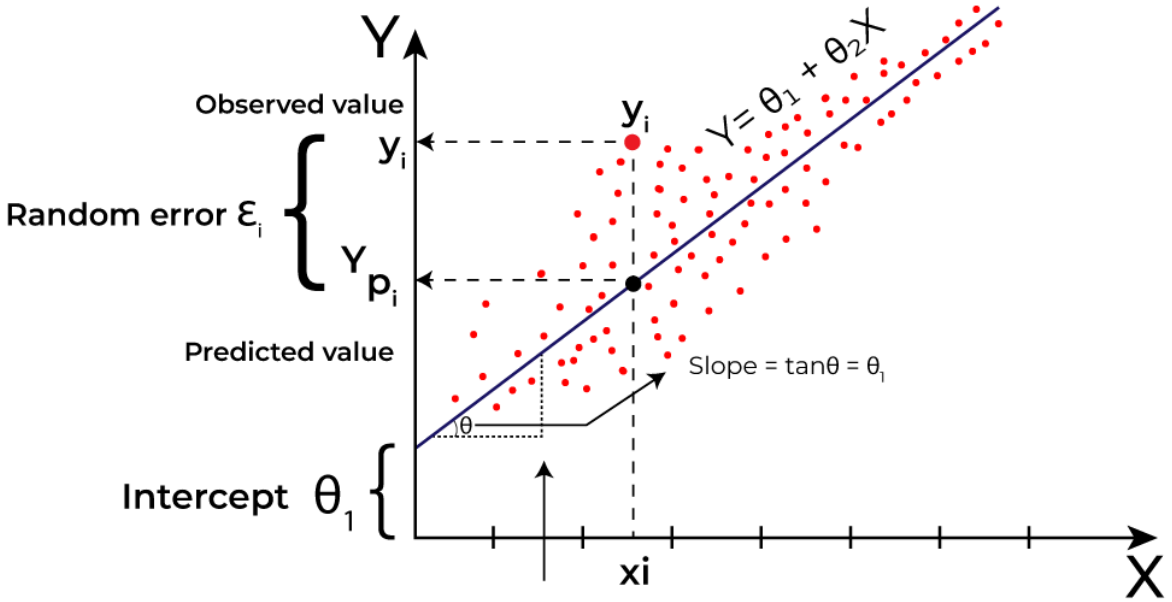
(nejčastěji podle metody nejmenších čtverců)

Trénování modelu
Trénování spočívá v nalezení optimálních koeficientů tak, aby se minimalizovala celková chybovost predikce, vetšinou metoda 80/20

Metriky pro hodnocení
1. MAE (mean absolute error) - průměrná hodnota rozdílů mezi predikovanou a reálnou hodnotou
2. MSE (mean squared error) - zjistí, jak daleko je jeho bod od přímky a poté podle délky vytvoří kostku

Přetrénování (Overfitting)
Může se stát, že náš model má nízkou chybovost v train datech, ale vysokou chybu na test datech - model je příliš složitý nebo málo train dat

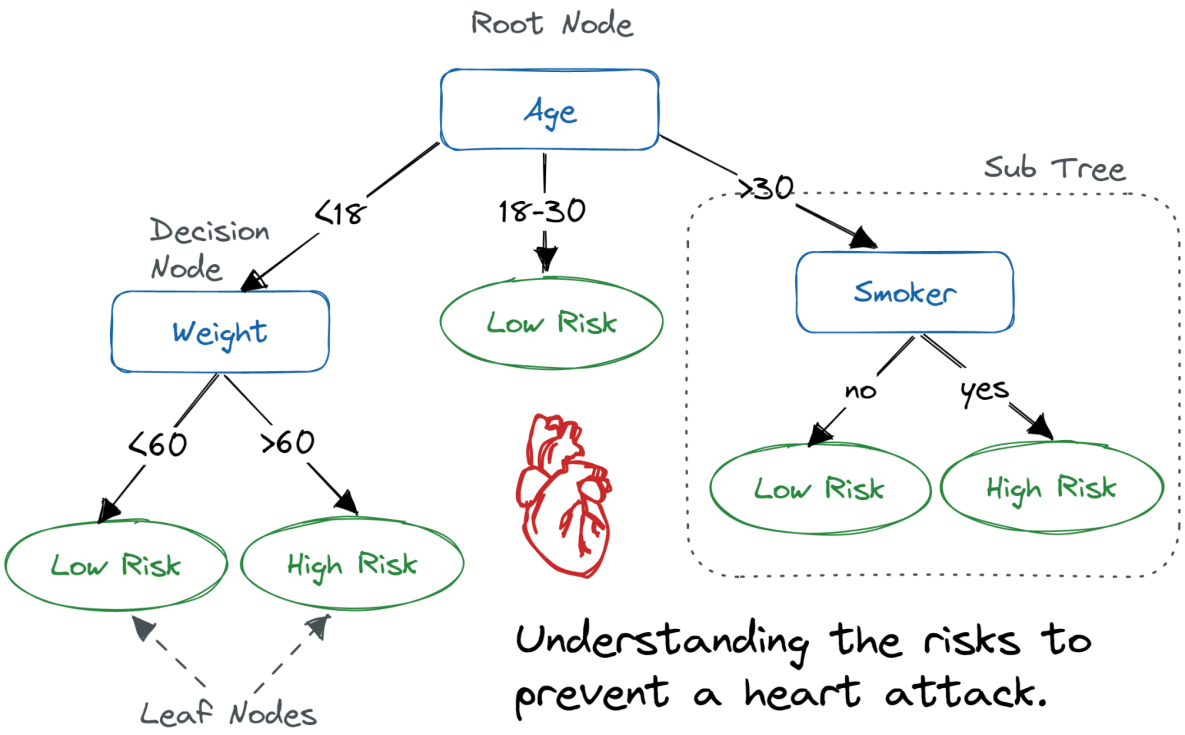
Podtrénování (Underfitting)
Opak přetrénování - model je příliš jednoduchý a nedokáže zachytit vztahy v datech




```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Načtení vestavěného datasetu (diabetes)
diabetes = datasets.load_diabetes()
X = diabetes.data  # Vstupní vlastnosti
y = diabetes.target  # Cílová hodnota (progrese diabetu)

# Rozdělení na trénovací a testovací data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Vytvoření a trénování modelu
model = LinearRegression()
model.fit(X_train, y_train)

# Predikce na testovacích datech
y_pred = model.predict(X_test)

# Vyhodnocení modelu
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
print(f"MAE: {mae:.2f}")
print(f"MSE: {mse:.2f}")
```

Rozhodovací stromy

Alternativní metoda regrese, která rozděluje prostor vstupních proměnných na části pomocí binárních podmínek. Na rozdíl od lineární regrese nevytváří matematickou funkci ale sérii rozhodnutí a dokazají zachytit i složitější nelineaŕní vztahy a interakce mezi proměnnými. Strom začíná kořenových uzlem obsahující všechna train data. V každém uzlu hledá nejlepší rozdělení dat. Predikce je poté uložena v koncových uzlech (listy). Snažíme se nalézt binární strom s minimální chybovostí.



Klasifikace
Klasifikace spočívá v určování na základě vstupních dat. Výstupem klasifikace je nějaká kategorie (barva, ano/ne….)

Princip
Například máme 10 hrušek a 10 jablek. Musíme počítači říci, co je jablko a co je hruška a řekneme mu například, že čím více červené, tím více je jablko, a čím více je šišaté, tím více je to zase hruška. Podle dat zjistíme, kde přibližně se nachází přechod mezi tím, kde je jablko a kde je hruška. Vždycky se stane, že některá jablka jsou zase zelená. Žádný algoritmus ovšem nemůžeme být naprosto přesný.

Klasifikační algoritmy
1. Metoda nejbilžsího souseda
    - neprovádí žádné učení, pouze ukládá data
    - při klasifikaci nového vstupu hledá nejbližší trénovací příklady a přiřadí mu nejčastější třídu mezi nimi
2. Logistická regrese
    - rozšíření lineární regrese pro binární klasifikaci
    - používá se pro připady spam/nespam, zdravý/nezdravý….
    - sigmoidální funkce
3. Rozhodovací stromy
    - existují stromy i pro klasifikaci

Hodnocení klasifikace
1. Správnost (accuracy) - kolik procent vzorků bylo zařazeno správně
2. Přesnost (precision) - Jak často je pozitvní predikce správná (Pokud předpovíme, že pacient je nemocný, jak často je to pravda?)
3. Pokrytí (recall) - Kolik skutečných pozitivních případů bylo správně identifikováno?
4. F-score - Harmonický průměr přesnosti a pokrytí

```
# Import potřebných knihoven
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt

# Načtení Iris datasetu
iris = datasets.load_iris()
X = iris.data  # Vstupní vlastnosti (délka a šířka kalichu a korunního plátku)
y = iris.target  # Cílové třídy (0 = setosa, 1 = versicolor, 2 = virginica)

# Rozdělení dat na trénovací a testovací sadu (70% trénink, 30% test)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Vytvoření a trénování modelu rozhodovacího stromu
model = DecisionTreeClassifier(max_depth=3)  # Omezení hloubky pro prevenci
přetrénování
model.fit(X_train, y_train)

# Predikce na testovacích datech
y_pred = model.predict(X_test)

# Vyhodnocení modelu
accuracy = accuracy_score(y_test, y_pred)
print(f"Přesnost: {accuracy:.2f}")
print("\nKlasifikační report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```