Python - Doporučení léčebné metody

Příprava prostředí

Pro správný běh této predikce je potřeba vytvořit virtuální prostředí, do kterého se nainstalují závislosti. Postupujte následovně:

```
# Klonovat repozitář nebo stáhnout .zip
git clone https://github.com/Bahamut731lp/DM
cd ./DM

# Vytvoření virtuálního prostředí
python -m venv env
# Pro windows, pro linux je příkaz 'source env/bin/activate'
.\env\Scripts\activate
# Nainstalování závislostí
pip install -r ./requirements.txt

# Spuštění aplikac
python main.py
```

Business Understanding

Cílem úlohy **Doporučení léčebné metody** je na základě údajů o pacientech určit, který lék by jim měl být předepsán. Cílem je tedy na základě předchozích záznamů klasifikovat pacienty do skupin podle toho, jaký lék jim byl předepsán. Tento přístup předpokládá, že léky byly pacientům podány správně a že všechny faktory, které předepsání léku ovlivňují, jsou součástí datasetu.

Data Understanding

K dispozici máme 200 záznamů a 7 sloupců. O jednotlivých pacientech víme následující údaje:

Sloupec	**Popis**
Age	Věk
Sex	Pohlaví
BP	Krevní tlak
Cholesterol	Hladina cholesterolu v krvi
Na	Hladina sodíku v krvi
K	Hladina draslíku v krvi
Drug	Lék, který byl pacientovi předepsán

Na základě Business Understanding je zřejmé, že cílovou proměnnou představuje sloupec **Drug**. Ostatní sloupce pak fungují jako potenciální prediktory. Nyní je nezbytné určit, které z těchto prediktorů jsou relevantní pro trénování modelu a které nejsou. K tomu jsem využil algoritmus pro odhad vzájemné informace, který umožňuje kvantifikovat míru závislosti mezi jednotlivými prediktory a cílovou proměnnou. Tento postup nám pomůže vybrat pouze ty proměnné, které skutečně přispívají k predikci, a tím zjednodušit model a zlepšit jeho výkonnost.

Z grafu v příloze 1 je patrné, že mezi pohlavím a užíváním konkrétního léčiva neexistuje žádná významná souvislost. Naopak, je zřejmé, že hladina sodíku v krvi a krevní tlak hrají klíčovou roli, protože tyto faktory jsou významně spojené s podaným léčivem.

V testovacích datech je však struktura mírně odlišná – místo samostatných sloupců *Na* (hladina sodíku) a *K* (hladina draslíku) se zde nachází sloupec s názvem Na_to_K, který představuje jejich poměr. Tento sloupec vznikl jako výsledek dělení hodnot sodíku hodnotami draslíku (*Na* / *K*). Aby bylo možné trénovací a testovací data smysluplně porovnávat a použít stejný model, bude nutné ve fázi přípravy dat sloupce *Na* a *K* v trénovacích datech sloučit do nového prediktoru Na_to_K. Důležitost tohoto prediktoru je vyobrazena v příloze 2.

Data Preparation

Prvním krokem bylo načtení a správné natypování dat. Pomocí knihovny pandas a metody read_csv() jsem vytvořil tzv. DataFrame - jedná se o strukturu, která reprezentuje dvourozměrná tabulková data. Pandas automaticky odvozuje datové typy, ovšem s nominálními daty si neuměl poradit. Prvním krokem bylo tedy přetypování vybraných sloupců na kategorická data.

Sloupec	Odvozený datový typ	Určený datový typ
Age	int64	int64
Sex	object	Categorical
BP	object	Categorical
Cholesterol	object	Categorical
Na	float64	float64
K	float64	float64
Drug	object	Categorical

Dalším krokem bylo spojit hladiny sodíku a hladinu draslíku do nové proměnné, `Na_to_K`. Důležitým krokem bylo také převedení textových kategorií do numerických, se kterými mohou modely pracovat. K tomu se v knihovně sklearn využívá třída LabelEncoder, která slouží k transformaci řetězců a čísla a zpět.

Modeling

K samotnému modelování jsem přistoupil komplexnějším způsobem – místo volby jednoho konkrétního algoritmu jsem se rozhodl vyzkoušet hned několik různých přístupů strojového učení a porovnat jejich výkonnost. Cílem bylo zjistit, který z modelů se nejlépe hodí pro daný úkol klasifikace léčiv na základě vstupních dat. Konkrétně jsem zvolil následující algoritmy:

- **K-Nearest Neighbors (KNN)** – Jednoduchý a intuitivní algoritmus, který klasifikuje vzorky na základě toho, jaké třídy mají jeho nejbližší sousedé. Výkon silně závisí na volbě hodnoty k a míře podobnosti mezi daty.
- **Random Forest** – Ensemble metoda založená na velkém množství rozhodovacích stromů, které jsou trénovány na náhodných podmnožinách dat. Výsledná predikce je určena hlasováním. Výhodou je robustnost a odolnost vůči přeučení.
- **Extra Trees Classifier** – Podobný princip jako Random Forest, ale s ještě větší náhodností při rozdělování uzlů ve stromech. Tento přístup často poskytuje rychlejší trénink a někdy i lepší generalizaci.
- **Gradient Boosting** – Pokročilá ensemble metoda, která staví modely postupně tak, aby každý další model opravoval chyby toho předchozího. Výsledkem je velmi výkonný, ale zároveň výpočetně náročnější model.
- **Multilayer Perceptron (MLP)** – Typ umělé neuronové sítě, který se hodí pro nelineární klasifikační úlohy. Využívá vrstvy neuronů propojených váhami, které se během učení přizpůsobují pomocí zpětné propagace.
- **Decision Tree** – Základní stromový klasifikační algoritmus, který dělí data do větví na základě hodnot atributů. Je snadno interpretovatelný, ale náchylný k přeučení, pokud není správně omezena jeho hloubka.

Jeden z hlavních problémů, na který jsem při vytváření modelů narazil, byla velmi nízká úspěšnost predikcí – pohybovala se kolem 18 %, což odpovídá zhruba náhodnému odhadu v rámci vícetřídové klasifikace, když máme 6 možných tříd léčiva. Po důkladném zkoumání se ukázalo, že problém nebyl ve vstupních datech ani v samotné struktuře modelu, ale ve způsobu kódování a zpětného dekódování cílové proměnné. Konkrétně šlo o to, že různé části kódu používaly různé instance enkodéru, což vedlo k nekonzistenci mezi trénovací a predikční fází. Jakmile jsem upravil kód tak, aby se pro kódování i dekódování používala jedna a ta samá instance, přesnost modelu dramaticky vzrostla – a to až na 92 %.

V rámci fáze modelování jsem se dále zabýval otázkou normalizace číselných atributů. K tomuto účelu lze ve frameworku scikit-learn využít například třídu `StandardScaler()`, která provádí standardizaci hodnot (tj. převedení na rozdělení se střední hodnotou 0 a směrodatnou odchylkou 1). Po provedení experimentů a vyhodnocení výsledků jsem ale zjistil, že modely bez normalizace dosahovaly mírně lepších výsledků – konkrétně o 2 až 3 procentní body vyšší přesnosti. Z toho důvodu jsem se rozhodl nadále pracovat s původními (nenormalizovanými) hodnotami.

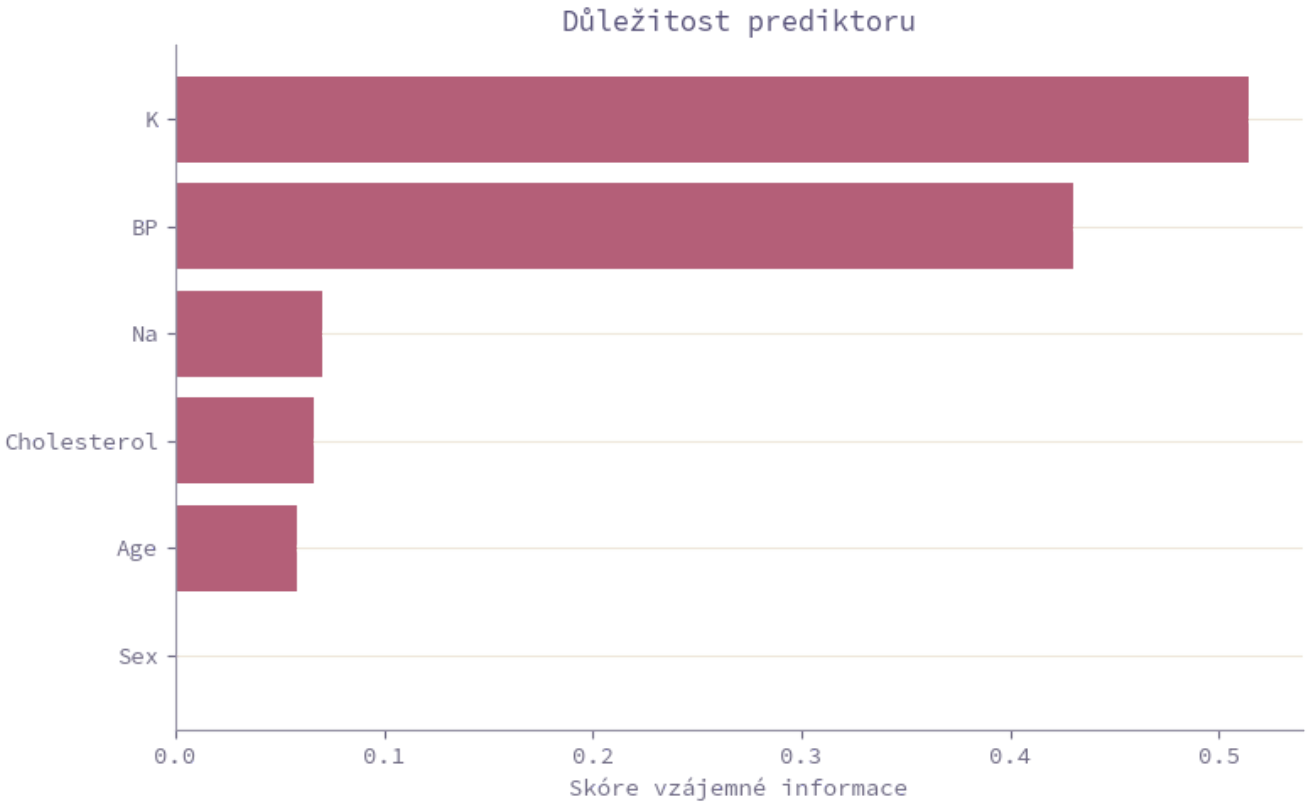
Evaluation

Pro vyhodnocení jednotlivých modelů byla použita především metrika přesnosti (accuracy) vypočítaná na testovací množině. Tato metrika poskytuje základní přehled o tom, jak často model správně klasifikuje záznamy. Kromě toho jsem využil funkci classification_report z knihovny scikit-learn, která pro každou třídu vypočítává další důležité statistiky – konkrétně precision (přesnost), recall (úplnost) a f1_score (harmonický průměr těchto dvou hodnot). Tyto metriky poskytují hlubší pohled na výkonnost modelu, zejména pokud je rozdělení tříd nevyvážené.

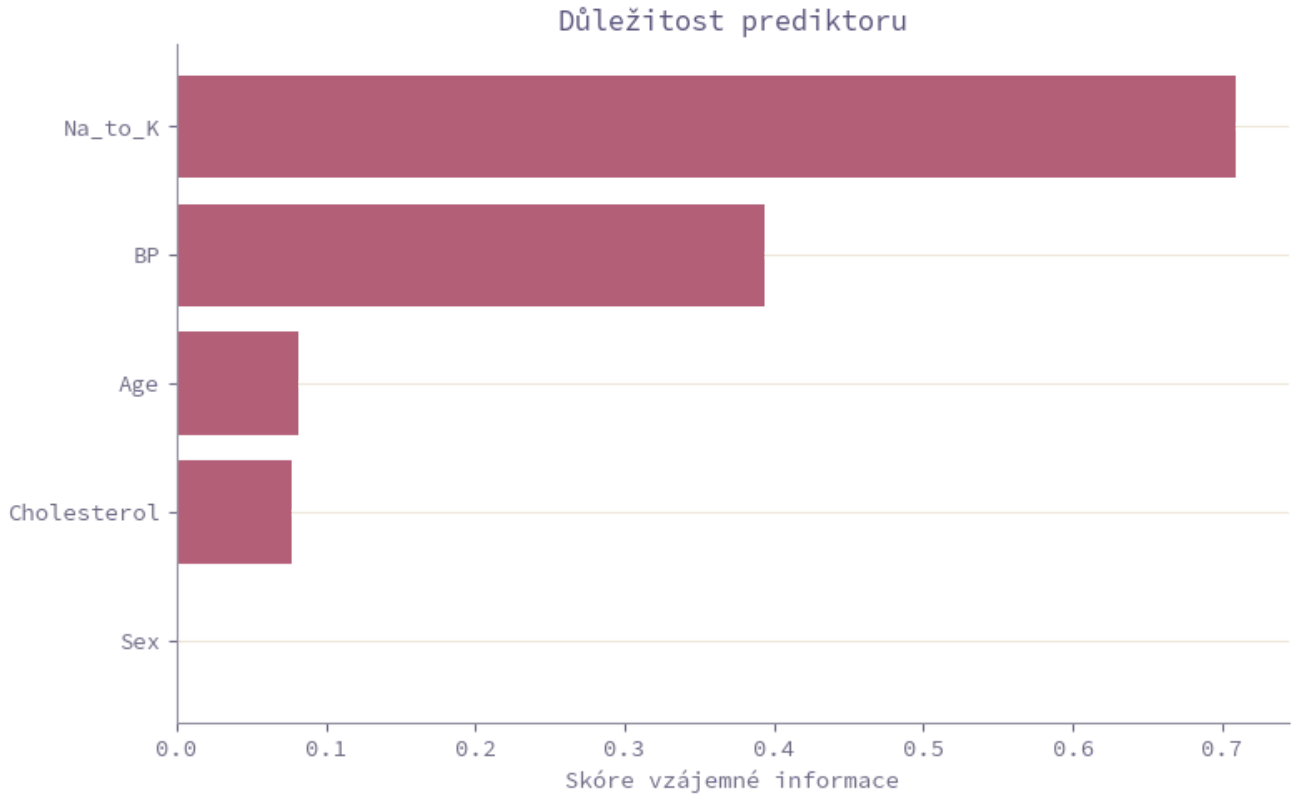
Pro snadnější interpretaci výsledků koncovým uživatelem aplikace generuje přehledný HTML report, který je uložen do složky report/. Tento report obsahuje tabulky s přehledným shrnutím přesnosti všech použitých modelů a podrobnými statistikami pro každou jednotlivou třídu. Uživatel si tak může snadno porovnat jednotlivé modely a učinit informované rozhodnutí o tom, který z nich je pro danou úlohu nejvhodnější.

Přílohy

Příloha 1: Důležitost prediktorů



Příloha 2: Důležitost prediktorů po spojení hladiny sodíku a draslíku



Příloha 3: Rozhodovací strom