Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Expertní sytém v Pythonu

Diplomová práce

Autor: Jan Horáček
Studijní obor: Aplikovaná informatika
Vedoucí práce: doc. RNDr Kamila Štekerová, Ph.D.
Hradec Králové

Prohlášení

Prohlašuji, že diplomovou práci jsem zpracoval samostatně a s použitím uvedené literatury.

Podpis:

V Hradci Králové 7.11.2021 Jan Horáček

Poděkování

Děkuji vedoucímu své diplomové práce paní doc. RNDr Kamile Štekerové PhD. a konzultantovi MUDr. Ondřeji Živnému za odborné vedení a cenné podněty a připomínky. Můj dík patří i všem, kteří mi poskytovali v průběhu zpracování mé diplomové práce svou odbornou i lidskou pomoc.

Anotace

Tato práce se věnuje expertním systémům a problematice umělé inteligencePopisuje základy expertních systémů, neuronových sítí, shrnuje využití Pythonu a v neposlední řadě popisuje webový framework Flask, který byl použit na vyvinutí prezenční vrstvy a backendu pro expertní systém. Expertní systém byl vytvořen pomocí frameworku Experta. Praktická část se věnuje návrh a implementaci expertního systému, který bude mít za úkol odhalit nemoci, které budou dostupné v bázi pravidel. Tato práce se bude věnovat predikci/detekci plicního emfyzému.

Anotation

Title: Expert system in Python

This thesis deals with Expert systems, respectively expert systems in Python. It describes the basics of expert systems, summarizes the use of Python and, finally, describes the Flask web framework, which was used to develop the presence layer and backend for the Expert System. The expert system was created using the Expert framework. The practical part is devoted to the design and implementation of an expert system, which will have the task of detecting diseases that will be available in the rules base. This work will focus on prediction/detection of the lung emphysema.

Obsah

1 Úvod

V dnešní době stále více pronikají informační technologie do zdravotnické sféry. Toto se například týká počítačové tomografie (CT). CT vyšetření je mnohdy časově náročné a velmi nákladné, avšak velkou výhodou tohoto vyšetření je snímkování pacienta v řezech, které mohou být zkompletovány do 3D modelu, kde je snazší hledat ohniska nemoci, či jiné relevantní znaky k dané nemoci. Naproti tomu rentgenové snímkování (RTG) je levnější a znatelně časově méně náročné. Ročně jsou po celém světě provedeny více něž dvě miliardy tohoto vyšetření. [10]

Identifikace nemocí ze snímků RTG je občas velice složitý proces. Jsou v dispozici různé metody, které zpřehledňují snímek RTG, avšak tyto metody vyžadují jistou přípravu. Dále je nutné zmínit, že zdravotník, respektive radiolog, který provádí zkoumání daného rentgenového snímku musí mít již zkušenosti, jelikož každý člověk je unikátní a daná nemoc na první pohled nemusí být znatelná, přičemž zdravotník bez zkušeností může tento projev přehlédnout.

Ročně podstoupí rentgenování velký počet obyvatel. Kvůli některým nemocem je nutné provést navíc magnetickou rezonanci nebo CT. Vyšetření CT představuje pro pacienta mnohem vyšší dávku radiace, nežli je tomu u RTG vyšetření, a to až 400krát. Tento markantní rozdíl není možné zanedbat a pokud není toto vyšetření krajně nutné, nevyužívat ho. U magnetické rezonance je jedním z problémů vyšetření pacienta s klaustrofobií a je tedy zbytečné tyto pacienty vystavovat vlivům a podmínkám, které by jim mohly způsobit jinou (psychickou) újmu na zdraví. Magnetická rezonance představuje pro pacienta velmi zdlouhavý proces od čekacích lhůt, až po samotné vyšetření, které může trvat řádově několik desítek minut.

Spojení neuronových sítí s expertním systémem se jeví jako dobrý způsob, jak snížit náklady na vyšetření i nepříjemnosti z pohledu pacienta. Nebylo by třeba pacienty vystavovat CT vyšetřením, pokud bychom pomocí vhodně navržené aplikace odhalili nemoc z pacientova RTG snímku a souvisejících údajů. Zkoumanou nemocí je plicní emfyzém.

Expertní systém je zamýšlen jako doplněk k neuronové síti, jelikož lze usuzovat, že dobře naučená síť rozpozná danou nemoc lépe nežli expertní samotný systém. Avšak na rozdíl od neuronové sítě, která se spoléhá pouze na obraz, se expertní systém zaměřuje na fakta, která byla vyplněna lékařem a pracuje tedy již nad jinou množinou dat a výsledek z neuronové sítě může podpořit nebo vyvrátit.

V této práci byl zvolen k vytvořen expertního Python a knihovna/enigne Experta. Knihovna Experta je inspirována pravidly v CLIPS**.** [2] Expertní systém by měl být schopen na základě výše zmíněných pravidel vyhodnotit ze vstupních informací, zdali pacient trpí zvolenou nemocí, či nikoliv.

Systém bude zpracovávat vstupy, které zadá lékař do formuláře společně s RTG snímkem. Tyto informace budou doplňující a rozšíří možnosti systému. V závislosti na poskytnutých informacích bude možné stanovit další obtíže, které by mohli pacienta postihnout, nebo zpřesnit procentuální šanci zastoupení dané nemoci.

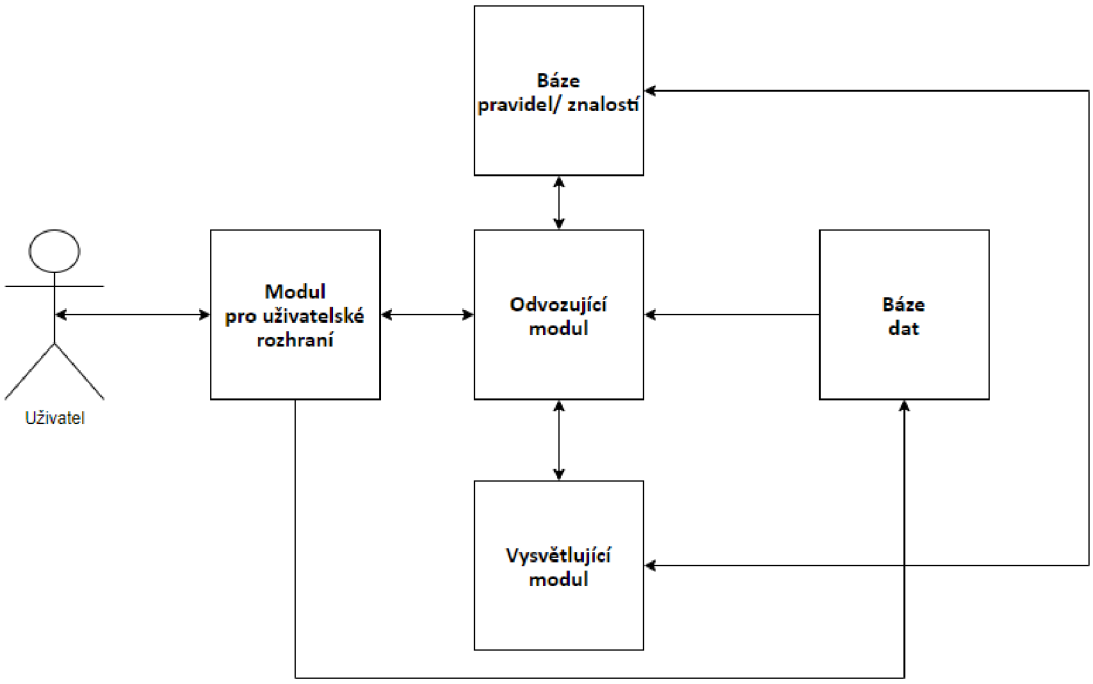
Hlavní výhodou tohoto řešení by bylo značně zkrátit čas pacienta strávený v nemocnici, či jiném zdravotnickém zařízení určeném pro kontroly. Pro medicínskou obec by to též mělo pozitivní dopad. Z RTG plic a vyplnění dodatečných vstupů, může být odhalena nemoc, která by jinak odhalena nebyla.

2 Teoretická část

2.1 Expertní systémy

Expertním systémem (ES) nazýváme systém nebo počítačový program, který simuluje rozhodování experta v dané oblasti. První systém/program, který můžeme považovat za expertní systém je systém DENDRAL. Byl vyvinut již v 60. letech 20. století, avšak dokončen byl až v 70. letech 20. století. Byl nazýván též heuristický DENDRAL. V začátcích byly expertní systémy vytvářeny diskutabilními praktikami. Například, pokud byl program vytvořen klasickým jazykem jako byl FORTRAN či PASCAL, byl následně přepsán do tzv. AI jazyku pro expertní systémy, tedy LISP nebo PROLOG. Toto vedlo k otázkám, kdy někteří analytici odmítali využití ES. [22]

Expertní systém se skládá z následujících komponent. [1]



Obrázek 1 Schéma ES

- **Báze pravidel/ znalostí** – je předpisem pro rozhodování a na základě těchto pravidel systém následně rozhoduje. Jedná se o souhrnný předpis znalostí experta, který v daném oboru působí. V této bázi nejsou uložena pouze všeobecná data, která jsou popsána například v učebnicích, článcích atp., ale jsou zde také zahrnuty přímo osobní znalosti a

zkušenosti daného experta. Těmito znalostmi a zkušenostmi se expert liší od začínajících odborníků v daném odvětví.

- **Báze dat** – jedná se o konkrétní případ dat, který by měl být aplikován na bázi pravidel/ znalostí a vyřešit daný problém.
- **Modul pro uživatelské rozhraní** – má za úkol poskytnou přehledný interface pro expertní systém i pro laického uživatele, který není v daném oboru expertem.
- **Vysvětlující modul** – modul, který dokáže popsat, jak systém k daném rozhodnutí došel a proč rozhodnutí učinil.
- **Odvozující modul** – (Inference engine) slouží pro odvozování, tedy dokáže zpracovat pouze ty informace z báze pravidel, které slouží pro řešení uživatelova problému.

Jedná se tedy o znalostní bázi expertního systému v daném prostředí. V dnešní době mohou být již implementovány prvky strojového učení, který postupem času zlepšuje výsledky expertního systému stejně tak jako praxe u běžného člověka. [11]

Expertní systémy dělíme z hlediska typologie do několika kategorií. Tyto kategorie rozřazujeme dle různých kritérií: [1][11]

Z hlediska charakteru řešených úloh rozlišujeme:

- **Diagnostický ES** – tyto systémy vyhodnocují a porovnávají předem definované hypotézy, z kterých vybírají hypotézu, která se nejvíce přibližuje reálným datům.
- **Plánovací ES** – systémy plánovacího typu předem znají stav počátku a stav konce. Pomocí kombinatorického generátoru generují možná řešení, která jsou omezena bází dat a znalostí. Úkolem je vyhledat optimální řešení k danému problému.
- **Hybridní ES** – hybridní typ ES vznikl spojením plánovacího ES a diagnostického ES. Využíván především pro monitorovací systémy.

- Podle míry obecnosti lze rozlišovat:
  - **Prázdný ES** – vyvinut pro diagnostické ES, neobsahuje bázi dat a znalostí.
  - **Problémově orientovaný ES** – prázdný ES, který je doplněn o bázi znalostí. Problémově orientovaný ES řeší pouze konkrétní problémové oblasti.
  - **Kompletní ES** – ES řešící konkrétní případ. ES obsahuje bázi dat a znalostí.

- Dalším kritériem je způsob reprezentace znalostí: **ES založené na pravidlech** – typický příklad řešení typu IF -> THEN. Pokud se v poskytnutých datech vyskytne určitá situace, ES provede předem definovaný úkon. Tímto způsobem lze vytvořit graf, kdy každý uzel odpovídá pravidlu (IF) a orientovaná hrana odpovídá tomu, co má systém provést (THEN).
- **ES rámcové** – informace o znalostech jsou vyjádřeny pomocí datových struktur neboli rámců. Rámce můžeme chápat či připodobnit k třídám v objektovém programování a následně dělit na objekty. Rámce tedy umožňují určité rozdělení a rozřazení dle nastavených preferencí.
- **ES založené na logickém programování** – znalosti vyjádřeny pomocí logických formulí, není vhodné využívat je v případě neurčitých znalostí. Důvodem je nemožnost logicky naprogramovat neurčitost. Jelikož logika odporuje neurčitosti. V logickém programování máme vždy určeno „kam se vydat dál“. Neurčitost by do takového chování vnesla chaos.

Dalším kritériem, které rozděluje ES, jsou generace. ES systémy dělíme do dvou generací. [1] [7]

2.1.1 ES a umělá inteligence

Pokud vycházíme z faktu, že ES spadá do oblasti umělé inteligence, pak můžeme tvrdit, že do základní problematiky ES patří: [14]

- **Neurčitost** – v reálném světě nefungují vždy přesná pravidla. Tedy například nemůžeme vždy vše rozřadit do různých tříd. Pro lepší představu mějme příklad silných malých aut. Nikde není definováno, jaké auto je silné a jaké je malé. Přesná definice není. Toto můžeme označit jako neurčitost, nemáme dozajista správné informace. Takováto informace je nepřesná, zkreslená či je zatížena nějakou chybou. Neurčitost je mnohdy modelována určitým číslem, které zastupuje míru přiřazení do určité skupiny.
- **Mechanismy řízení** – tyto systémy zabezpečují využívání znalostí v ES. K těmto mechanismům řízení můžeme přistupovat dvěma směry:
    - **Prohledávání prostoru řešení** – vycházíme z řešení daných úloh.

- **Nevycházíme z řešení úloh** – technika obsahuje další rozpis podtechnik, které jsou každá specifická svým využitím informací, zapisování pravidel, změnou modelu atp.

- **Znalostní reprezentace**

  - **Znalost** – bývá velmi často chybně definována. Některé definice srovnávají znalost = informace. Toto však je chybný úsudek. Jelikož znalost popisuje spíše osvojenou zkušenost, či vlastnost.

  - **Modularita báze znalostí** – jedná se o důležitý prvek ES. Tento prvek zajišťuje aktualizaci báze znalostí tím, že doplňuje a třídí nabyté znalosti do jednotlivých existujících bází. Pokud dojde ke změně báze, ES pracuje pouze s částí báze, nikoliv s celou bází. Oproti tomu, pokud dojde k hledání, je nutné bázi prohledat celou.

2.1.2 Oblasti využití expertních systémů

V tomto odstavci jsou uvedeny příklady dalších možných využití expertních systémů. Pokrytí je opravdu široké, od financí až po medicínu.

Mezi finanční expertní systémy, které byly využity v praxi, řadíme například FINEVA, INVEX, FAME. Tyto systémy byly využívány především v 90 letech 20 století v bankách, marketingu a službách. [26]

Je možné pozorovat, že vědecké články věnující se expertním systémům, které opravdu byly v provozu a které byly využívány, jsou z 90 let 20. století až přelomu milénia. V dnešní době se články zaobírají především spojením expertních systémů a neuronových sítí. A to tak, aby došlo k vyloučení nevýhod každého ze systémů, kdy se expertní systém neponaučí z chyby, kterou provedl a neuronové sítě mají typicky jedno zaměření (identifikace obrazu, rozpoznání obrazu atp.).

2.1.3 ES ve zdravotnictví

Expertní systémy ve zdravotnictví byly již využívány dříve. Autor článku *„Expert system with an embedded imaging module for diagnosing lung diseases“* uvádí hlavní výčet známých expertních systémů, které se v historii využívaly. [14] Jejich hlavním úkolem bylo pomáhat diagnostikovat nemoci a pomáhat při výuce. K nejznámějším zástupcům ES, které spadají do oboru medicíny, patří například:

- **MYCIN** – tento systém byl využíván na identifikaci bakterií. Zároveň dokázal doporučit léky v závislosti na pacientově váze. MYCIN obsahoval přibližně 600 pravidel. Dotazoval se lékaře jednoduchými otázkami ANO/NE v některých případek pokládal též „vypisovací“ otázky. MYCIN se nikdy nedostal do „ostrého“ provozu. Důvodem bylo hned několik faktorů. Jedním z nich byla etická otázka, která je dodnes aktuální. V případě, že systém vyhodnotí chybně a pacient dojde k újmě na zdraví, kdo bude zodpovědný za tuto újmu? Vývojář systému? Lékař? Jelikož na tuto otázku nebylo možné zodpovědět v minulosti ani nyní, nedošlo k nasazení MYCINU. Dalším důvodem byla nutnost zadávat uživatelský vstup. V této době však nebyly vyvinuty a rozšířeny ještě PC (personal computer) a tak systém pracoval v raných fázích ARPAnetu. Systém MYCIN ukázal sílu a možnost vyhodnocovat podávání léčiv pomocí ES. [24][22]
- **PXDES** – expertní systém využíván pro predikci stupně a typu rakoviny plic. Tuto predikci vyhodnocoval z RTG snímků, kde v závislosti na stínech odhadoval stupeň rakoviny plic / pneumokoniózu (zaprášení plic).
- **CaDet** – uváděný jako jeden z nejlepších expertních systémů na identifikaci ranných stádií rakoviny. Docílí toho tak, že zkoumá epidemiologické a klinické atributy daného pacienta a snaží se vyčíslit a zobrazit lékaři vzory, na které by se měl více zaměřit. [13]
- **Internist** – ES Internist je uváděn jako jeden z největších ES systémů. [24] Důvodem je pokrytí přibližně 80% obecné medicíny a popis přibližně 750 zdravotních poruch.

Výše zmíněné expertní systémy patří mezi nejznámější a zároveň nejúspěšnější expertní systémy v oboru medicíny, které zároveň souvisí s touto prací. Expertní systémy jsou většinou využívané tam, kde je jasně definován problém a kde můžeme se strojovou přesností určit požadované vlastnosti, a tak i zautomatizovat daný proces.

Aktuální využití ES ve zdravotnictví je méně časté než například u neuronových sítí. Avšak stále se zde vyskytují systémy, které kombinují tyto dva směry. [23] Tedy kombinují ES a NS do jednoho celku jako celistvého systém. Důvodem je klasifikace nebo také předzpracování a klasifikace. Tímto je myšleno, že pokud se budeme zaobírat právě RTG plic, je vhodné odeslat RTG snímek na zpracování do NS, která je pro zpracování obrázku velice výhodná. Výsledek z NS je následně odeslán na validaci do ES.

Systémů tohoto typu, které by se právě zaobíraly spojením NS a ES pro detekci nemocí z RTG snímků není mnoho. Vědecké práce v aktuální době se zaměřují především na NS a jejich využití pro identifikaci COVID-19.[31][32][33][34] [35]. Avšak některé práce se věnují filtraci a rozpoznávání či identifikaci chronických obstrukčních plicních nemocí (CHOPN) v závislosti na datech o pacientovi. Data vyplní lékař daného pacienta pomocí dotazníku/ formuláře. Výstupní data z dotazníku jsou odeslána do neuronové sítě, která počítá jednotlivé váhy a předzpracovává vstupy. Výstup z této neuronové sítě je odeslán do expertního systému, který pracuje na typické bázi IF-THEN a dle výstupních parametrů z NS určí celkový výstup/ identifikuje daný typ CHOPN. [29] Dále jsou zde práce, které se věnují pouze expertnímu systému a RTG snímky neberou v potaz. Směrodatná data jsou pro ně pouze data, která zadal lékař. [30]

Dalším problémem, který brání v zapojení těchto systémů v reálném využití, je etická otázka „Kdo bude mít zodpovědnost za chybu? Programátor? Lékař?“. Systémy mohou fungovat jako podpůrné a jejich výsledek bude podpořen a potvrzen expertem.

Expertní systém je definován dle pravidel a parametrů. Tento fakt lze využít například pro přefiltrování nemoci a následně odeslat požadavek přímo do daných neuronových sítí, které budou detekovat přímo danou nemoc. Vznikly též práce zabývající se ES, které by usnadnili lékařům práci, případně by pacienta „vyslechly“ před samotnou návštěvou lékaře samotného. Některými z těchto poměrně nových systémů jsou například Polymyalgia Rheumatic Expert System, který provádí diagnózu a následně terapeutické cviky. [15] Dalším takovýmto zástupcem je ES pro diagnózu endokrinních obtíží a jejich následná léčba. Autor využil k řešení této problematiky jazyk Java, respektive JESS, JESS

neboli Java Expert Systém Shell zaštiťuje funkce, pravidla, fakta a procedury. Dle slov autora byl výsledek expertního systému uspokojivý a slibný. [17]

2.1.4 Výhody a nevýhody ES

Domnívám se, že výhody ES jednoznačně převažují nad jeho nevýhodami. Jelikož ES může být funkční stále (nepociťuje únavu atp.), může být chápán jako podpůrný systém pro rozhodování. ES je rychlý a dokáže přesně odvodit, jak k závěru došel. Dokáže též ušetřit finanční prostředky, které by bylo nutné zaplatit expertovi lékaři v rámci konzultací při tvorbě systému.

Na druhé straně, pokud srovnáme experta a ES, ES na rozdíl od lidské bytosti, která dokáže uvažovat, nedokáže ES reagovat na nečekané situace, nedokáže také zapojit intuici, jelikož žádnou nemá. Jedná se o strojový program. Vychází jenom a pouze z pravidel a vědomostí, kterých nabyl. Pokud nedojde ke korektnímu přenosu znalostí od experta lékaře přes inženýra systému do výsledného ES, může docházet k fatálním chybám určování ES. V některých případech může být údržba ES velmi nákladná, toto však není pravidlem.

Tabulka 1 Srovnání Expert vs ES porovnává možnosti experta s možnostmi expertních systémů.

Expert	**ES**
Pomíjivost (stáří, emoce)	Stálý
Obtížné předávání znalosti	Přenosný
Obtížné zdokumentovat	Snadný ke zdokumentování
Nepředvídatelný (emoce)	Konzistentní
Nákladný	Méně nákladný

Tabulka 1 Srovnání Expert vs ES

Tabulka 2 Popis Účastníků ES zobrazuje účastníky, kteří se podílí na tvorbě ES a zároveň popisuje role jednotlivých účastníků.

Účastník	Role
Expert	Osoba, která je expertem v daném odvětví, které bude ES řešit. Jedná se o základní předpoklad úspěšného ES
Inženýr znalostí	Inženýr znalostí (Knowledge engineer) se stará o převod a zápis znalostí do ES/ počítačového programu
Koncový uživatel	Uživatelé, kteří ve výsledku využívají ES. Jedná se například o naprosté laiky nebo též začínající mediky.

Tabulka 2 Popis Účastníků ES

Tabulka 3 Porovnání Konvenčního systému a ES přehledně srovnává konvenční systémy s ES a poukazuje na jejich výhody a nevýhody.

Konvenční systém	**ES**
Znalosti a procesní mechanismus jedna jednotka.	Báze znalostí a procesní mechanismus. Jsou odlišné
Program nedělá chyby. (Nebereme v potaz chybu v kódu)	ES může chybně vyhodnotit.
Systém plně funkční až po zhotovení.	ES by měl být optimalizován a být schopen operovat i s malým množství pravidel.
Běh krok za krokem.	Běh je prováděn logicky a heuristicky (báze znalostí).
Vyžaduje plnou informaci.	ES nevyžaduje plnou informaci. (implementace)

Tabulka 3 Porovnání Konvenčního systému a ES

Většina citovaných autorů se shoduje na tom, že by nemělo dojít k nahrazení experta lékaře expertním systémem, avšak systém by měl pomoci identifikovat nemoci. Příkladem může být využití v rozvojových zemích, kde je nedostatek zdravotnického personálu a tyto státy se musí spoléhat na humanitární pomoc. [19][17] Dále uvádí, že ES může vyhodnocovat v procentech v závislosti na poskytnutých datech a odůvodňují to tak, že expert, v tomto případě lékař, neuvádí zastoupení nemoci procentuálně, ale spíše slovně odůvodní, proč si myslí, že daná nemoc je přítomna. [19]

2.2 Neuronové sítě

Neuronové sítě (dále také NS) jsou v dnešní době značně rozšířené a zasahují téměř do každého odvětví. Jejich všestrannost dovoluje vzájemné prolnutí.

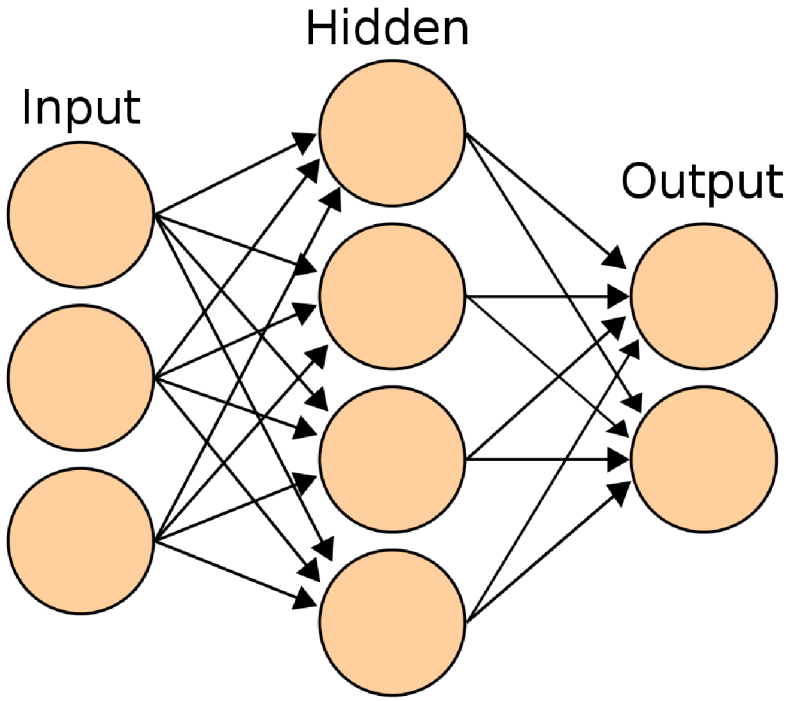
Neuronové sítě vycházejí z poznatku o neuronech a nervových sítích živých organismů. Schopnosti NS rozřazujeme následujícím způsobem: [12]

- Zobecňovat
- Učit se
- Řešit úlohy
- Schopnost extrahovat a reprezentovat data

Základní variantou NS je perceptron, který reprezentací jednoduchého modelu dopředné sítě (bude vysvětleno dále v této kapitole). Model jednoho perceptronu je možné využít pouze na řešení lineárně separovaných problémů. Modely, které obsahují pouze perceptron, respektive jednovrstvou neuronovou síť, dokáže vyčíslit logické operace AND nebo OR, avšak XOR nikoliv. Uvedený problém vedl k názoru, že na řešení tohoto problému je nutné mít vícevrstvou NS. Marvin Minsky svým způsobem pozastavil vývoj téměř na dvě dekády, protože v dané době nebyl formulován algoritmus pro vícevrstvé sítě a dle jeho tvrzení takovýto algoritmus nebude možné nalézt, pokud vezmeme v potaz komplikovanost sítí. [12] Tvrzení vyvolalo pozastavení masového zájmu o NS, avšak někteří badatelé pokračovali ve výzkumu a v roce 1986 představili odvozený algoritmus pro učení těchto sítí. Nazývá se algoritmem zpětného šíření chyb, tzv Backpropagation algoritmus (BA). BA opět přivedl NS zpět do středu zájmu. [12] BA se využívá k učení podle očekávaného výsledku. Během učení algoritmus porovnává výstup z neuronové sítě s očekáváným výstupem. Následně je provedena úprava vah, aby došlo k minimalizaci odchylky, která vznikne po vypočítání mezi aktuálním výstupem z NS a očekávaným výstupem z NS.

Avšak algoritmus BA s sebou přináší problém, tzv. overfitting, nebo také přeučení. Přeučení se snadno pozná na validačním setu. Jedná se o situaci, kdy síť se zdá být velice dobře natrénována, avšak pokud NS předložíme validační set, selže. Důvodem je nemožnost zobecňovat. Přeučení lze tedy chápat tak, že NS se naučí pouze testovací sadu, ale pokud ji předložíme dříve neznámý snímek, vyhodnocení bude náhodné. Přeučení lze předejít dostatečně velkým datasetem a včasným přerušením trénovaní. K tomuto můžeme využít funkce, které nabízejí jednotlivé frameworky. [8]

NS jsou tvořeny neurony. Neurony zpracovávají vstupní signály a produkují signály výstupní. Komplexnější NS jest topologické uspořádání neuronů, které spolu komunikují pomocí orientovaných hran. Například u dopředných sítí se jedná o jednostranně orientované hrany, tedy každý neuron má pouze vstup a výstup a signál se šíří jedním směrem od vstupu po konečný výstup sítě. Pro jednodušší představu tohoto typu sítí je uveden Obrázek 1 Dopředná NS [6], který tento typ znázorňuje:



Obrázek 2 Dopředná NS [6]

Obrázek 2 Dopředná NS [6] zobrazuje výše zmíněné orientované hrany. Toto propojení je vždy mezi předchozím a následujícím neuronem, nikdy mezi neurony jedné vrstvy. Tento typ neuronové sítě můžeme nazývat též jako paralelní distribuovaný dynamický systém obsahující neurony, které pracují jako samostatné jednotky. Toto zajišťuje určitou robustnost a odolnost vůči poškození sítě. Spoj této sítě je ohodnocen určitou vahou. Při učení sítě se tyto váhy modifikují, v některých případech se modifikuje i vnitřní uspořádání. [12]

2.2.1 Typy neuronových sítí

Existuje mnoho typů neuronových sítí. Vybrány byly pouze nejznámější a běžně používané sítě, které jsou základem pro přiblížení problematiky neuronových sítí.:

- **Dopředné sítě**

Jedná se o nejběžnější typ NS. Konstruovány jsou způsobem výstup z jednoho neuronu je vstupem do jiného neuronu. U tohoto typu sítí lze pro učení využít BA algoritmus.

- **Rekurentní sítě**

Rekurentní sítě obsahují na rozdíl o dopředných sítí smyčky. Smyčky zajišťují to, že výstup z neuronu je zároveň i jeho vstupem. Na rozdíl od dopředných sítí je výpočetně mnohem náročnější. Je tedy nutné zvážit použití těchto sítí.

- **Konvoluční sítě**

Jedná se o jakýsi poddruh dopředných sítí. Konvoluční sítě jsou navíc obohaceny o tzv. konvoluční filtry, které umožňují snadnější detekci vzorů v obraze, proto se tyto sítě používají především na identifikaci obrazu.

2.2.2 Typy učení

Typy učení neuronových sítí nazýváme adaptačními algoritmy. Obecně můžeme říci, že učení probíhá obdobným způsobem, jako je tomu u člověka. Na známých případech se naučíme určitou věc, kterou pak dokážeme aplikovat na předem neznámý případ. Tedy například, pokud má neuronová síť rozpoznávat psa a kočku, je nutné jí poskytnout dostatečný počet zástupců psů a koček, aby po předložení obrázku, který se nenacházel v testovací sadě, dokázala určit, zdali se jedná o psa či kočku. Při učení je nutné vstupní obraz upravit, abychom ho zbavili nepotřebných informací. Toto se provádí například zmenšením rozlišení obrazu. Tedy není nutné, aby NS měla kompletní informaci o celém obrazu, ale stačí aby byla patrná informace o zkoumaném objektu. Způsoby, kterými lze učení provést, je vícero. Zde budou uvedeny jen ty nejznámější. Učení s učitelem, učení bez učitele. [8]

- **Učení s učitelem** – trénovaní probíhá iterativně. Algoritmus předkládá prvky z trénovací množiny NS. Následně zjistí odchylku na předložený vstup a upraví váhu neuronu. Učení probíhá v epochách, tedy po předložení veškerých dat z trénovací

množiny. Toto učení může trvat jednotky až tisíce epoch v závislosti na řešeném problému.

- **Učení bez učitele** – učení, které probíhá bez vnějšího zásahu, síť třídí data pouze na základě shlukování množin. Je řízena pouze vstupními daty a jak jsou rozděleny. Nemá informaci o správnosti zařazení.

2.2.3 Využití neuronových sítí

Neuronové sítě jsou využívány dnes a denně, například pro identifikaci a zařazení oblečení do kategorií u některých větších e-shopů jako je Zalando) NS se též používají na identifikaci lidí z kamerových záznamů, toto můžeme pozorovat především ve větších městech, které touto technologií disponují. V neposlední řadě jsou NS též využívány v některých vozidlech, kde se zároveň učí na aktuálním dění, provozu, situacích atp. (např. Tesla).

Jelikož se trend NS dostává čím dál tím více a více do popředí, můžeme nalézt na platformách zabývajících se touto problematikou využití všech možných směrů. Lze tak říct, že pokud vytváříme síť identifikující nějakou věc, je téměř jisté, že již někdo tuto síť vytvořil před námi. Avšak nikdy nedosáhneme stejných výsledků, protože datasety, na kterých se učení provádí, vstupní parametry, transformace, předpříprava datasetů je různorodá. Kombinací je nezměrné množství.

NS pronikly do zdravotnictví a jsou vedeny různé výzkumy, které pospojují NS do řetězce, kdy obraz prochází každou NS a výsledek je vrácen jako procentuální nález každé nemoci, která je zkoumána. Přesnost těchto NS se značně liší. Některé sítě se velice blíží v rozpoznávání nemocí znalostem odborného lékařského personálu bez dlouholetých zkušeností z praxe. [20][21]

Vzhledem k aktuálnímu dění a výskytu nemoci COVID-19, se nově vznikající články věnují především identifikaci nemoci COVID-19 či jiných plicních nemocí kterými jsou například pneumonie, nemoci spadající do CHOPN. [31][32][33][34] [35]. V těchto článcích je úzká spolupráce mezi lékaři a IT odborníky, kteří danou síť vytváří. Lékaři dodávají datasety, které jsou „olablované“ a připravené k využití pro učení, validaci a testování. Následně je pak proveden test na několika typech sítí. Zejména autoři využívají ResNet, DensNet anebo VGG. Veškeré uvedené sítě jsou sítěmi konvolučními. [38]

Jelikož je velká pozornost věnována právě neuronovým sítím, je snížený zájem právě o kombinaci ES + NS. Avšak spojení těchto dvou „systémů“ do jednoho, by mohlo přinést uspokojivější výsledky v rámci dalších vstupních dat a tím pádem větší základny znalostí o nemoci a následně detailnějšímu rozhodování při identifikaci/ predikci.

2.3 Python

2.3.1 Obecný popis

Python je programovací jazyk, který můžeme nazývat interpretovaným, objektově orientovaným (dále také OOP) a v neposlední řadě interaktivním. Zahrnuje vše potřebné pro vývoj aplikací, ať už těch klasických, či webových. Dokáže obsluhovat výjimky, moduly a samozřejmě také třídy, které jsou velice důležité v OOP. Silnou stránkou Pythonu je jeho relativní všestrannost, s čímž souvisí jednoduchá použitelnost či využitelnost systémových volání. Funkce, které značně zjednodušují a urychlují psaní kódu v Pythonu. Tento jazyk je dále možné rozšiřovat pomocí jazyku C nebo C++. Python je též přenositelný, avšak je zde nutné brát ohled na to, zdali daný kód poběží na Unixových systémech, například Linuxu či Windows, jelikož při systémových voláních jsou určité rozdíly, které je nutné brát v potaz. Dále je nutné též zvolit správnou knihovnu pro import. [3] Na rozdíl od jiných programovacích jazyků, obsahuje Python pár výjimek, které částečně usnadňují a částečně stěžují zápis, pokud programátor je zvyklý programovat například v Javě. Prvním rozdílem je odsazování. V Pythonu odsazování hraje velkou roli, jelikož metody jsou zakončené dvojtečkou nikoliv otevírající složenou závorkou. Toto neplatí jen pro metody, ale také pro IF, FOR atd. Neméně důležité je odsazení, aby překladač věděl, že daný blok spadá pod metodu, IF, či FOR viz Kód 1 Formát kódu Python. Druhou rozdílností je soubor app.py. Tento soubor nazýváme Python scriptem, který je následně spouštěn. Script by měl splňovat

```
try:
    response = os.environ.get('URL_TO_NN'), data=form.rtg_scan.data)
    for r in response:
        if r.get("emphysema"):
            result = get_by_id(ResultOfTest, id=returned_id)
            result.emphysemaNN = r.get("emphysema")
            result.dateOfEnd = datetime.utcnow()
            baseService.update()
```

Kód 1 Formát kódu Python

konvence definované v PEP (Python Enhancement Proposals). Tato doporučení určují například formátování (odsazení, počet mezer, atd). Script může obsahovat pouze metody, třídy, či jiné prvky. Pokud chceme využít některé z prvků definovaných v daném scriptu v jiném scriptu, musíme pomocí importu definovat cestu k tomuto prvku (metoda, celý script, třída).

Python byl dříve používán především pro vědecké účely, statistické výpočty atp. Později však přinesl možnost tvorby tříd a začal též více pronikat do odvětví webových aplikací, kde si prorazil cestu díky své jednoduchosti a díky frameworkům, kdy mezi nejznámější patří například Django nebo Flask. [3] Python není využíván pro tvorbu webových aplikací v takové míře jako například Java, PHP, C#. Tyto zmíněné programovací jazyky jsou rozšířenější především z důvodu jejich většího povědomí v rámci programátorů a například u PHP, které bylo používáno pro tvorbu webů hned ve svém počátku. Java a C# slouží především se svými frameworky .NET a Spring k zástupcům tvorby podnikových aplikací pro velké firmy. Python byl tedy v ústraní a jeho zastoupení v tomto odvětví není tak silné jako jiné jazyky, avšak jeho zastoupení časem roste jako u jiných jazyků. Růst má na svědomí také větší poptávka po jiných jazycích, než je Java nebo C#.

Django na rozdíl od Flasku, je více robustní a nabízí mnohem více offshelve řešení. Django si lze přímo stáhnout před vytvořenou kostru aplikace, kde je již zahrnuto velké množství konfigurací. U Flasku je ražena spíše idea, pokud danou věc potřebujeme, tak si ji napíšeme. To znamená, že sice též nabízí kostru ke stažení, ale není zde značná konfigurace v porovnání s Djangem. Důvod je prostý, Flask je light weight nebo také mikro framework a Django je full stack webový framework. [37][36]

Uvedeno do kontextu. Pokud máme vytvořenou neuronovou síť a budeme chtít mít možnost dotazovat se z různých zařízení typicky z mobilního zařízení nebo z webového prohlížeče, vytvoříme si aplikaci založenou na Flask frameworku a vytvoříme v ní API, abychom se na tuto neuronovou síť mohli dotazovat. Toto funkční řešení nasadíme na funkční prostředí. Samozřejmě není zakázáno vytvářet plnohodnotné aplikace též ve Flasku, avšak na tuto formu využití je vhodnější využít framework Django.

2.3.2 Flask Framework

Flask Framework patří mezi nejpopulárnější tzv.light weight Python WSGI webové frameworky. [36] Jeho popularitu dosvědčuje též nástroj Google Trends kde je možné po vyhledání pozorovat jeho vzestup. WSGI znamená Web Server Gateway Interface. Toto vyjadřuje specifikaci, která určuje komunikaci mezi aplikací a webovým serverem a též komunikaci mezi aplikací samotnou. [4]

Flask obaluje Werkzeug a Jinja. Werkzeug a Jinja jsou základní knihovny, které v této práci využíváme. Werkzeug je knihovna zaštiťující WSGI, tedy zajišťuje detaily WSGI a zároveň poskytuje využívání struktur a paternů, které usnadňují vývoj. Jinja umožňuje zápis kódu ve frontendových stránkách. Jedná se tedy o tzv. template engine pro Python. Jeho syntaxe je snadno uchopitelná pro zápis a zároveň též pro pochopení, viz ukázka Kód 2 Jinja Syntaxe.

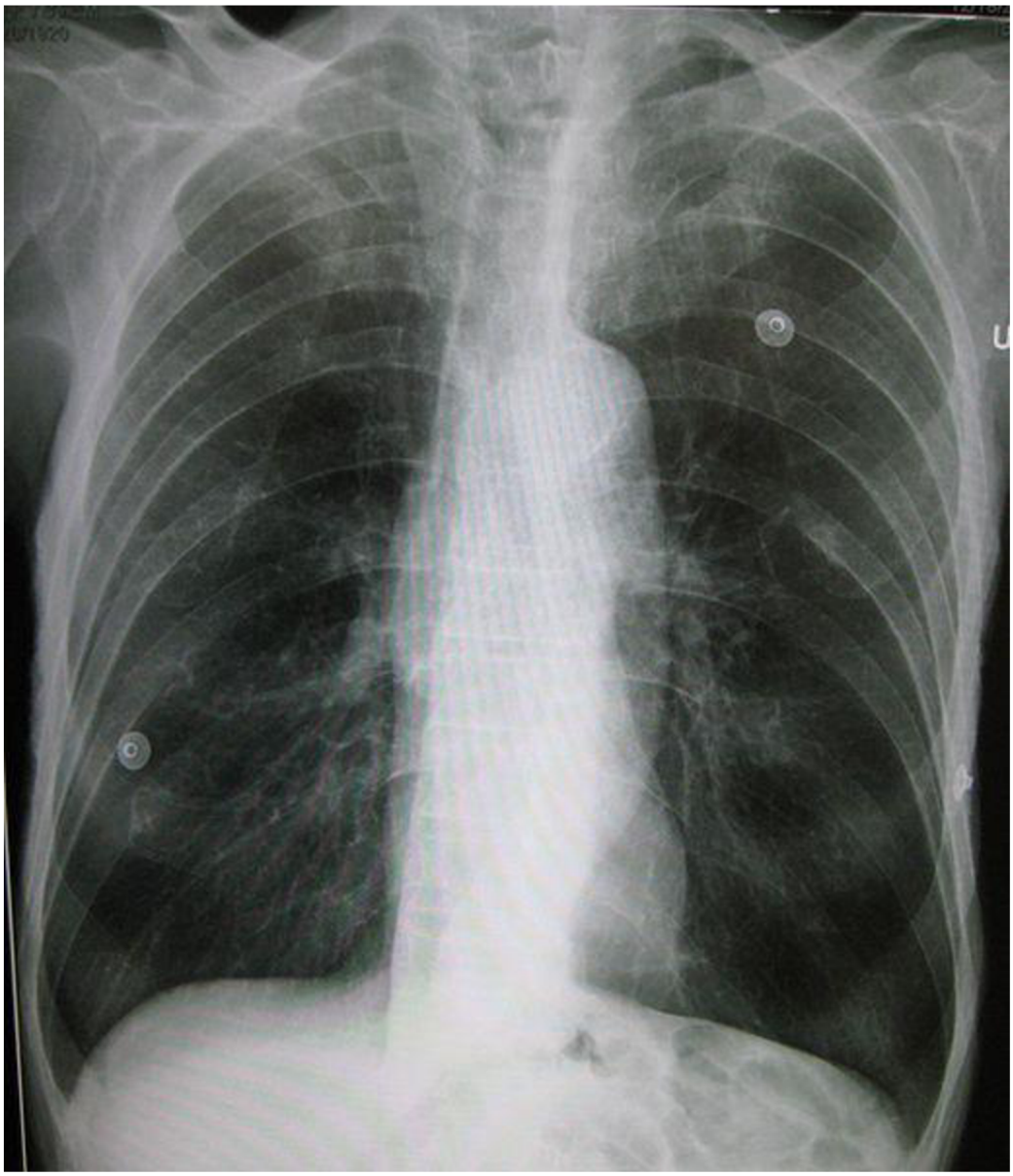
```
{% extends "layout.html" %}
{% block body %}
  <ul>
  {% for user in users %}
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
  </ul>
{% endblock %}
```

Kód 2 Jinja Syntaxe

2.4 Plicní emfyzém

Kapitola se věnuje popisu a vysvětlení nemoci plicního emfyzému, který bývá laiky často označován jako emfyzém. Samotné označení emfyzém není korektní, jelikož emfyzém označuje rozšíření tkáně ve spojitosti s nahromaděním vzduchu. Plicní emfyzém označuje nahromadění vzduchu v plicní tkáni. Jiným druhem emfyzému je kožní emfyzém, kdy je vzduch nahromaděn v kožní tkáni. Tato práce se věnuje plicnímu emfyzému, který je znám též pod pojmem rozedma plic. Plicní emfyzém postihuje především kuřáky a osoby v pokročilém věku, nebo naopak mladé osoby trpící deficitem alfa-1-antitrypsinázou, který způsobuje destrukci jak stěn průdušek tak i plicních sklípků. Jedním z identifikátorů, které plicní emfyzém doprovází je rozedmutí hrudního koše. Rozedmutí hrudního koše se projevuje v úrovní horní části hrudi, kdy osoba trpící plicním emfyzémem má nepřirozeně roztažený hrudní koš. Důvodem je právě nahromadění vzduchu v horní části plic, kde dochází k pomalejšímu odbourávání/ zpracování vzduchu na rozdíl od spodní části plic. Tento stav je možné identifikovat pomocí RTG snímku, a to zploštělou bránicí. Zploštělá bránice nemusí být ihned pozorovatelná, a tak radiologové či lékaři provedou manuální měření, které se provádí tak, že v „srpečcích“ bránice vztyčí dva body, které následně propojí a zkoumají, zdali vytvořená přímka splňuje definované parametry. Obrázek 3 RTG pacienta s plicním emfyzémem znázorňuje plicní emfyzém. Zde je možné pozorovat rozedmutí plic a zploštělou bránici, dále je možné pozorovat v horních částech tmavá místa, která znázorňují zavzdušnění neboli místa, která obsahují nadměrné množství vzduchu. [28]

Jelikož je plicní emfyzém chronická nemoc, jeho léčení je zdlouhavé. Pacient musí přestat kouřit, dále jsou pacientovi předepisovány lázeňské pobyty a v neposlední řadě potřebné léky.



Obrázek 3 RTG pacienta s plicním emfyzémem [27]

3 Praktická část

3.1 Návrh řešení

Mým záměrem je vytvořit experní systém, propojitelný s neuronovu sítí. Systém má usnadnit rozhodování lékaře při diagnostikování plicního emfyzému.

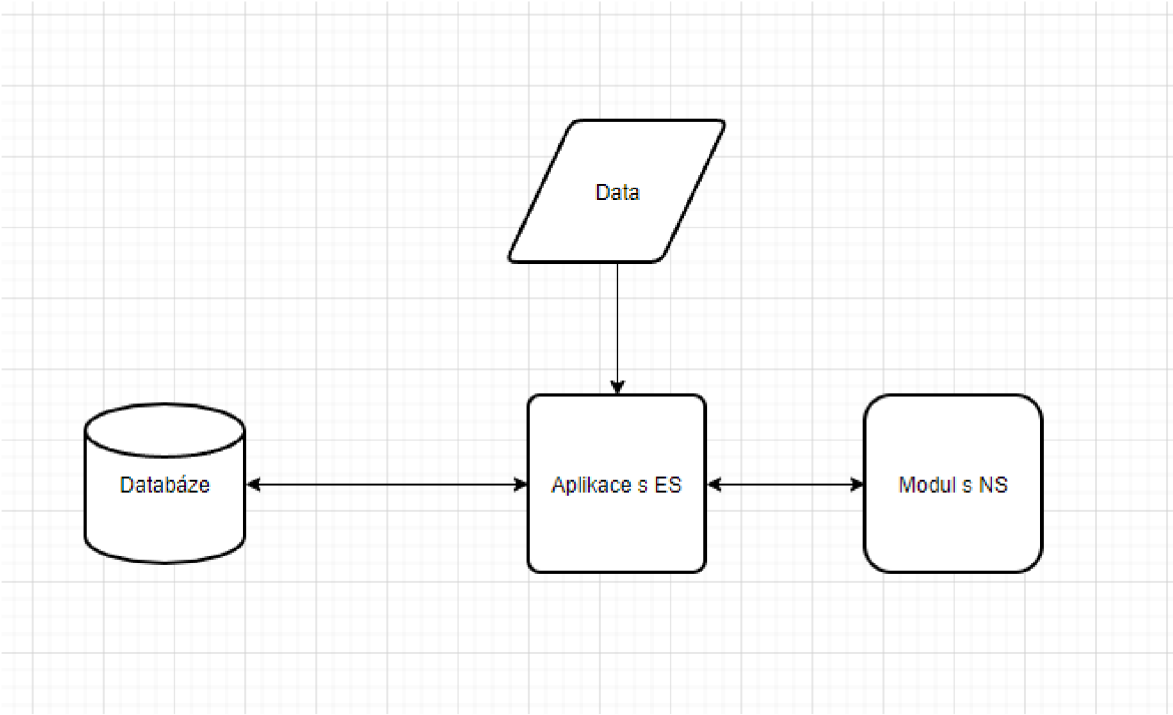
Implementace bude formou webové aplikace, která by měla obsahovat přehledné a intuitivní uživatelské API, kde bude možné nahrát RTG snímek(y) a metadata o pacientovi.

V rámci této aplikace bude též jisté předzpracování údajů do formy, která bude předána neuronovým sítím pomocí volání API metodou POST. Vyhodnocení bude vráceno uživateli, zároveň se však vyhodnocení uloží do databáze, která bude uchovávat výsledky kvůli historizaci dat a možného zpětného dohledání vyhodnocení pro daného pacienta. Tato databáze by měla obsahovat pouze údaje o pacientovi, které zadal lékař pomocí formuláře v aplikaci.

Typ aplikace bude klasický přístup MVC, kdy bude uživateli zobrazeno view, data se budou zpracovávat v controllerech. Zároveň zde bude modul, který bude komunikovat na server, kde budou připraveny neuronové sítě. V rámci odpovědi, která bude nejspíše ve formátu JSON, by měla přijít procentuální šance výskytu dané nemoci, respektive s jakou přesností můžeme říct, že je zde přítomna požadovaná nemoc. Výsledek bude dále předán expertnímu systému potažmo přímo uživateli v závislosti na zvolené předvolbě (více v kapitole 3.3).

Data se do databáze budou ukládat v momentně, než budou předána uživateli. Forma uložených dat by měla být následující: Entita obsahující ID uživatele, následně zde budou sloupce, kdy názvy těchto sloupců budou zrcadlit nemoci. V těchto sloupcích budou uvedeny jednotlivé výsledky, které byly zaslány pomocí REST API do aplikace jako odpověď. Dále tato tabulka bude obsahovat sloupec textového typu, kam bude ukládán obsah JSONu, který bude vrácen z expertního systému.

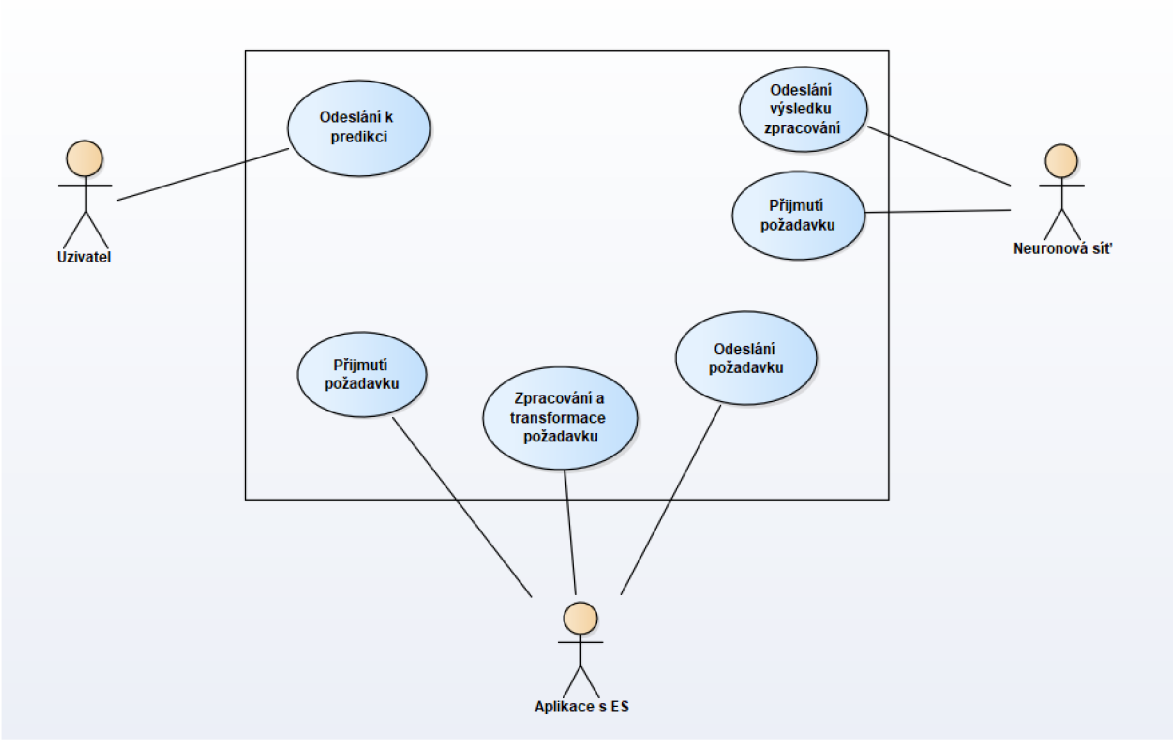
Pro účely expertního systému v tomto případě poslouží systém vytvořený v Pythonu, kdy se nejedná o expertní systém CLIPS, avšak o jeho port/ kopii přenesenou do prostředí Pythonu. [2] Následující obrázky znázorňují předběžný návrh aplikace a jejích modulů.



Obrázek 4 Schéma aplikace

Obrázek 4 Schéma aplikace znázorňuje návrh budoucího schématu aplikace a jak bude rozdělena do modulů. Středem je webová aplikace, která bude komunikovat na ostatní moduly. Aplikace s ES znázorňuje vytvořenou aplikaci s prezentační vrstvou a přístupem uživatele k vytvořeným záznamům o pacientech, nemocech atp. Modul s NS znázorňuje samostatný modul nezávislý na Aplikaci s ES. Toto nabízí možnost měnit NS na pozadí, bez nutnosti změny v aplikaci, pouze pomocí změny URL v konfiguraci na serveru. Databáze znázorňuje úložiště dat s pacienty, nemocemi a výsledky z predikcí. Data znázorňují vstup uživatele do systému/ aplikace.

Obrázek 5 Use Case aplikace zobrazuje případy užití. Je zde možné pozorovat čtyři aktéry. Jedním je uživatel, ostatní jsou systémy, které umožní běh aplikace. Možnosti uživatele jsou dostačující pro účel tohoto systému, predikce nemoci. Aplikace se stará o zpracování dat, ES a odesílání požadavků na NS.



Obrázek 5 Use Case aplikace

Báze pravidel bude implementována pomocí frameworku Experta a vazeb v databázi. Toto umožňuje snadnější zápis báze pravidel a výsledné vyhodnocení. Viz Kód 4 Definice pravidla, Kód 3 Definice faktu a Kód 5 Definice tříd pro fakty.

```
@Rule(Symptoms(MATCH.symptoms),
      SymptomsToDisease(=MATCH.all_symptoms_for_disease,
MATCH.disease),
      TEST(lambda symptoms, disease: increment(symptoms, disease)))
def detect_disease_0(self, symptoms, all_symptoms_for_disease,
disease):
```

Kód 4 Definice pravidla

```
@DefFacts()
def init_emphysema_facts(self):
    yield PtfFev1VerySevere(maxValue=49, minValue=35)
    yield PtfFev1Severe(maxValue=59, minValue=50)
    yield PtfFev1ModeratelySevere(maxValue=69, minValue=60)
    yield PtfDlcoVerySevere(maxValue=39, minValue=0)
    yield PtfDlcoModerate(maxValue=59, minValue=40)
    yield PtfDlcoMild(maxValue=79, minValue=60)
    yield PulseOxymetry(minPulseOxymetry=95)
```

Kód 3 Definice faktu

```
class PtfDlcoVerySevere(Fact):
    maxValue = Field(int, default=39)
    minValue = Field(int, default=0)
class PtfDlcoFromPatient(Fact):
    pass
```

Kód 5 Definice tříd pro fakty

3.2 Návrh báze pravidel a odvozujících pravidel

3.2.1 Popis báze pravidel

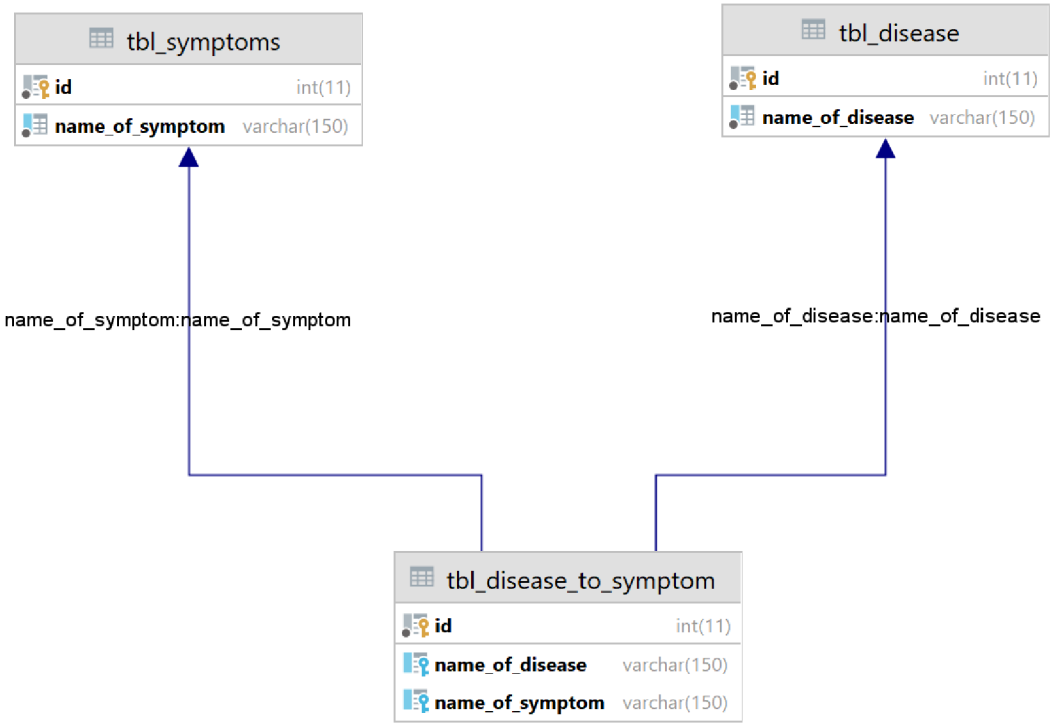
Z důvodu použití frameworku Experta pro Python je nutné upravit návrh pravidel, jelikož složitost na tvorbu pravidel je z důvodu práce s třídami, respektive rámci vyšší. Zároveň bude moci uživatel některá pravidla měnit (například symptomy k nemoci). V taktovém případě bude muset být pravidlo obecnější nežli u jasně specifikovaných pravidel, která budou nadefinována přímo a bez možnosti změny od uživatele. Pouze v kódové části.

Vzhledem k této obecnosti byla vytvořena báze pravidel pro symptomy a nemoci pomocí tabulek v databázi, pomocí schématu vazeb nemoc : symptomy. Vazba nám zaručí možnost obecného popisu pravidel pro nemoc a následně její identifikaci. Nastavení více nemocí umožní predikci více nemocí nežli jen plicního emfyzému (nebude implementováno v rámci této práce, avšak je zde ponechán prostor pro budoucí rozšíření). Bude však omezena v rámci nastavených pravidel uživatelem, jelikož vnitřní pravidla jsou modifikovatelná pouze přímo v kódové části aplikace. Tedy bude moci predikovat pouze v závislosti na symptomech.

Vnitřní pravidla přímo definovaná v aplikaci vychází z jasně definovaných jednotek například pro spirometrii, okysličení krve, váhy, výšky atp. Tato pravidla jsou neměnná. Jsou zde definované milníky, podle kterých můžeme rozhodnout, zda se jedná o kritickou hodnotu, normální hodnotu, případně pokud lze toto rozdělení více rozpadnout na mírné atp., budou jednotky takto uzpůsobeny a rozpadnuty. Báze těchto pravidel je uvedena v Python scriptu expertClass.py. V tomto scriptu je možné pozorovat, že každá informace je definována třídou. Tyto třídy jsou rozděleny dle závažnosti (závažné, normální atp.). Každá třída obsahuje minimální a maximální hodnotu, v případě měřitelných jednotek číslem, pokud se jedná o hodnotu ano/ne (true/false), je uvedena ta hodnota, která splňuje zastoupení pro danou nemoc. Tedy například jedním z parametrů je „kuřák“. Plicní emfyzém mohou mít též pacienti, kteří nejsou kuřáci, avšak pacienti, kteří kuřáci jsou, mají větší tendenci k tomuto onemocnění, kvůli změnám na plicích. V takovém případě bude tedy hodnota nastavena na true.

Na rozdíl od jiných ES, je vhodné zde definovat též třídy, které budeme inicializovat a vkládat do deklarací při inicializaci ES. Důvodem je lepší orientace ve faktech a snadnější práce s těmito fakty v rámci rozhodování. Tyto třídy budou mít prázdné tělo a naplněny

budou při inicializaci ES reálnými daty od pacienta. Báze pravidel se tedy odlišuje od návrhu, který je běžně využíván v CLIPS, avšak je nutné mít bázi pravidel nastavenou tímto způsobem, abychom dosáhli výše zmiňované univerzálnosti a všestrannosti pro pozdější využití.



Obrázek 6 Databázová struktura pravidel

Jelikož je nutné měnit určitá pravidla za běhu systému, případně vytvářet nová pravidla, je nutné nejdříve vytvořit definici nemoci a následně přidat symptomy k dané nemoci. Každá nemoc má vytvořen svůj vlastní záznam pro symptom, tedy je možná duplicita symptomů v tabulce symptomů. Databázová struktura spojení tabulek nemoci a symptomy viz Obrázek 6 Databázová struktura pravidel.

Pravidla je možné upravovat tím, že si zvolíme symptom pro danou nemoc, který lze smazat a vytvořit nový. Jelikož symptomy se liší, není nutné mít možnost úpravy. Tento fakt nám umožňuje lépe pracovat s daty a jejich úpravami.

3.2.2 Popis neměnných metadat

Níže jsou uvedeny parametry, které je možné zadat při vytváření pacienta a zároveň při odesílání dat do NS a ES. Některé z parametrů nejsou povinné, jiné povinné jsou. Povinné parametry jsou označeny hvězdičkou (*). Parametry BMI a FEV/CV jsou vypočítávány na základně jiných vstupních jednotek a označené vlnovkou (~).

- *** Věk –** decimální jednotka, která vyjadřuje stáří pacienta. Parametr věk je důležité získávat, protože usnadňuje následnou identifikaci plicního emfyzému v závislosti na tom, zdali je pacient kuřák a trpí deficitem alfa-1-antitrypsinázou
- *** Výška –** decimální jednotka určující délku pacienta
- *** Váha –** decimální jednotka určující hmotnost pacienta
- *** Kuřák/Krabičkoroky –** pokud je pacient kuřák, je nutné vyplnit též parametr krabičkoroky (krabičky za den krát počet let, jak dlouho pacient kouří). Díky krabičkorokám je možné určit o jak silného kuřáka se jedná.
- **Deficit alfa-1-antitrypsináza –** Jedná se o vzácnou nemoc, která se vyskytuje již u velice mladých lidí a může napomoci potvrdit výskyt plicního emfyzému.
- **Okysličenost krve (Pulzní oxymetrie) –** procentuální zastoupení kyslíku v krvi, kdy normální hladina se pohybuje nad 95 %. Parametr je velice obecný a je vyžadován spíše jako doplňující informace, jelikož nemocí, které souvisí s nedostatkem okysličenosti krve je více druhů.
- **FEV (FEV1) –** *„objem vzduchu vydechnutý s největším úsilím za 1. sekundu po maximální nádechu“* [25], udáváno v litrech
- **FCV –** *„maximální objem vzduchu, který lze po maximálním nádechu prudce vydechnout“* [25], udáváno v litrech
- **~ FEV1/CV –** *„jednotka, která má zastoupení v procentech“* [25] a je dopočítávána pomocí rovnice:

$$\left(\frac{\text{FEV}}{\text{FCV}}\right) * 100$$

Jedná se o tzv. Tiffeneanův index, který poukazuje na míru dechového postižení v procentech. Tabulka 4 Rozložení Tiffeneanova indexu uvádí jednotlivé rozřazení dle závažnosti Tiffeneanova indexu.

Procento předpokládané hodnoty FEV1/CV	Výsledek
80 % nebo více	normální
70 % – 60 %	mírně abnormální
59 % – 50 %	mírně až těžce abnormální
49 % – 35 %	těžce abnormální
Méně než 35 %	velmi těžce abnormální

Tabulka 4 Rozložení Tiffeneanova indexu

- **DLCO** – difuzní kapacita plic pro oxid uhelnatý. Tabulka 5 DLCO procentuální redukce znázorňuje DLCO a jeho kritické hodnoty

DLCO redukce	Výsledek
75 % nebo více až do 140 %	normální
74 % – 60 %	mírně snížené
59 % – 40 %	mírně až těžce snížené
Méně než 39 %	velmi těžce snížené

Tabulka 5 DLCO procentuální redukce

- **~ BMI** – index tělesné hmotnosti, který je vypočítáván na základě tělesné hmotnosti (v kilogramech) a výšky (v centimetrech) pacienta. BMI je vypočítáno tímto vzorcem:

$$\frac{\boldsymbol{hmotnost(kg)}}{\left(\frac{\boldsymbol{výška(cm)}}{\boldsymbol{100}}\right)^{\boldsymbol{2}}}$$

BMI	Hodnocení (pro dospělé)
méně než 18,5	Podváha
18,5 až 24,9	Normální hmotnost
25 až 29,9	Nadváha
30 a více	Obezita

Tabulka 6 Krajní hodnoty BMI

3.2.3 Popis odvozujících pravidel

Odvozující pravidla, budou navržena tak, aby se zkoumala od nejvíce obecného pravidla k nejméně obecnému pravidlu, což lze nazvat procházení stromem. Tedy nejdříve bude kontrolováno, zdali koresponduje počet symptomů se symptomy, které uživatel zadal k pacientovi v souvislosti s nemocí, kterou uživatel vybral při odesílání do predikce. Pokud ano, bude nemoc zařazena do možných adeptů na shodu. ES však nekontroluje pouze symptomy. Kontrolují se zde další vstupní parametry, jimiž jsou měřitelné jednotky ze spirometrie (FEV, FVC, DLCO), dechovou frekvenci, okysličenost krve, zdali je pacient kuřák a s tím spojené tzv. krabičkoroky, věk, alfa-1-antitrypsináza, výška, váha a BMI. Dle požadavků nebude nutné vždy zadávat kompletní data, viz výčet výše. Zadávat kompletní data je irelevantní, jelikož ne všichni lékaři mají přístup k daným datům konkrétního pacienta. Pokud by program používal obvodní lékař, bude zadávat pouze základní parametry

jako věk, výška, váha, dechová frekvence, nebo zdali je pacient kuřák a s tím spojené krabičkoroky. Lékaři však nebude blokováno zadat ani další jednotky ze spirometrie či jiné parametry, budou však na první pohled skryta pod tlačítkem *„Zobrazit pokročilé nastavení“* a jejich vyplnění je nepovinné. Tím dojde k odstínění konkrétnějších parametrů. Pokročilé nastavení by bylo vhodné například pro plicní laboratoře, kde je možné využít pacientovi záznamy, případně pacienta rovnou vyšetřit na spirometrii. Neméně důležitým a pro ES velice přínosným parametrem je alfa-1-antitrypsináza. Svoji důležitost představuje především v tom, že je možné určit, zdali se neuronová síť může mýlit či nikoliv. Určení probíhá díky znalosti, pokud je člověk mladý a zároveň trpí alfa-1-antitrypsinázou a NS vyhodnotila pacienta s vysokým procentuálním zastoupením plicního emfyzému, můžeme předpokládat, že se síť nemýlí. Avšak pokud budeme mít stejný případ a pacient nebude trpět alfa-1-antitrypsinázou lze usuzovat, že NS mohla tento konkrétní případ odhadnout chybně, protože plicní emfyzém se vyskytuje především u starších jedinců, toto však není pravidlem jsou zde zohledňovány zbylé prvky vstupních parametrů. Výsledek ES však není závislý pouze na tomto jednom parametru, a tak celý proces ES může následně vyhodnotit výsledek z NS jako pravdivý.

Po inicializaci, ES prochází definovaná pravidla a zkoumá, zdali jednotky splňují či nesplňují danou hranici, případně zdali se neshodují. Pokud dojde ke shodě, dojde k provolání metody, která náleží danému pravidlu viz Kód 4 Definice pravidla. Po vykonání této metody je vytvořen nový Fakt, který určuje, že ES splňuje tuto podmínku, prošel daným pravidlem a našel shodu. S tímto nově vydefinovaných faktem dále ES pracuje a používá ho pro kombinaci daných měřitelných jednotek. Engine Experta, který umožňuje vytváření faktů, při vytvoření stejnojmenného faktu původní fakt nesmaže či nenahradí, ale vytvoří fakt nový. Tento způsob je velice výhodný, jelikož můžeme bez starostí vždy vložit fakt stejného typu, ale s různými parametry. V tomto případě se jedná o fakt Prediction(), který obsahuje nalezené shody, jež jsou definovány pomocí řetězce znaků. Řetězce znaků jsou následně kontrolovány v rámci pravidel. Engine Experta, který je v této práci využíván pro predikci je nastaven tak, že prochází každý možný vstup. V závislosti na tom, jaké podmínky jsou splněny, zvolí systém danou cestu či nikoliv.

Hlavními rozhodovacími prvky pro ES byly použity funkce z Experta Enginu. Těmito metodami jsou TEST (), AND (), OR (). Metody AND () a OR () vykonávají klasické porovnání výsledků, tedy AND je splněno tehdy, když jsou splněny všechny podmínky vložené do této funkce. OR je splněno, pokud alespoň jeden ze vstupních parametrů je

pravdivý. Vstupy do těchto funkcí jsou pouze Fakty, které jsou též výtvorem Experta Enginu. Vstupem do této metody může být opět metoda AND, OR, či jakákoliv jiná metoda z Experta Enginu. Metoda TEST () testuje vstupní parametr pomocí lambda výrazu viz. ukázka Kód 4 Definice pravidla. TEST metoda je však závislá na vstupním parametru. Vstupním parametrem je proměnná, která byla dříve „namatchována“. „Matchování“ pomocí MATCH slouží pro získání atributů daného faktu. S atributy lze dále pracovat jak v prostředí pravidla (@Rule) tak i v metodě, jelikož se tento „namatchovaný“ atribut stává též atributem vstupním do dané metody, která naleží danému pravidlu viz Kód 4 Definice pravidla.

3.3 Návrh rozhraní a funkce aplikace

Samotné rozhraní je obsahuje jasně definovaná tlačítka, která jsou na první pohled dobře viditelná. Svou roli zde hraje barevnost samotné aplikace, která uživatele přímo směruje na vybraná místa. Dalším prvkem, který usnadní práci s aplikací je šrafování tabulek, to do jisté míry zpřehledňuje tabulky, které se v systému nacházejí.

Uživatel může vytvořit záznam o pacientovi. K vytvořenému záznamu o pacientovi bude nutné zadat údaje jako jsou údaje o jeho váze, výšce, jiných nemocech a také informace o životosprávě (kuřák/nekuřák). Po úspěšném vytvoření záznamu se bude moci uživatel prokliknout do detailu tohoto pacienta.

Po prokliknutí bude zobrazen detail pacienta se všemi jeho informacemi. Zároveň zde bude uvedena tabulka s výsledky z předchozích měření. Pokud nebude k dispozici žádný výsledek pro daného pacienta, bude zde hláška *„Pro daného uživatele neproběhla žádná predikce“*. Dále na této stránce bude uživatel moci odeslat pro daného pacienta snímek pro predikci. Uživatel bude mít na výběr ze dvou možností. Odeslat pouze snímek pro predikci na NS, anebo odeslat snímek pro NS a ES současně. Tyto dva požadavky se liší ve formě odeslání dat. Při odeslání dat do NS proběhne pouze odeslání snímku pacienta. Pokud se uživatel rozhodne odeslat snímek do NS a ES bude odeslán nejen snímek, ale také veškeré informace o pacientovi, metadata, které budou podkladem pro rozhodování v ES.

Požadavek na predikci bude probíhat synchronně, tzn. odeslání požadavku na server a následně musí uživatel čekat na vyhodnocení. Výsledek predikce bude v závislosti na jejím typu obsahovat datum požadavku na predikci, včetně času zahájení a času dokončení. Dále zde bude procentuální odhad NS, zastoupení dané nemoci v odeslaném RTG snímku. ES bude vyhodnocen slovně, respektive bude v závislosti na pravidlech sestaven slovní výstup s doporučením. Vzhledem k postupnému skládání bude výstup značně slovně omezen na pouhá slovní spojení. Pokud uživatel bude chtít odeslat další predikci bude mu to umožněno, ale bude nucen znovu vložit RTG snímek a celý proces se bude opakovat.

Na domovské obrazovce bude zobrazen stránkovaný seznam pacientů. Přehled disponuje ikonami pro zobrazení detailu pacienta, aktualizaci pacienta, smazání pacienta. Pro přímý přístup na výsledky z NS a ES je možné využít tlačítko „Zobrazit“.

3.4 Vyhodnocení snímku a metadat

Samotné rozhodování bude realizováno pomocí Experty (Engine pro ES v Pythonu). [2] Experta byla vybrána především pro jednoduchost zápisu a přehlednost zápisu. Dalším kritériem byla jednotnost kódu v jazyku Python, která zajistila plynulý vývoj aplikace.

Následně bude aplikace, respektive ES, procházet jednotlivá pravidla a bude se v závislosti na vstupních informacích, které jsou odeslány z aplikace do ES ve formě metadat, rozhodovat, zdali pacient trpí danou nemocí, či nikoliv.

Dále bude proveden dotaz na neuronovou síť. NS by měla být nezávislá na aplikaci s ES. Nezávislostí je umožněna změna NS v případě potřeby, pokud bude dodržen formát, ve kterém se vrací predikovaná data. Komunikace s NS bude probíhat pomocí metody POST, kdy bude zaslán obrázek/RTG ve formátu jpg nebo png. V závislosti na obrázku/RTG bude zpět odeslána odpověď v daném formátu JSON. Po přijetí vyhodnocení požadavku z NS dojde k zapracování výsledných hodnot (procentuální odhad predikce dané nemoci) do vstupu pro ES. ES vyhodnotí vstup z formuláře pro pacienta a vstup od NS, zpracuje data a výsledek uloží do tabulky s výsledky. Pokud se nepodaří identifikovat nemoc, bude pole pro

```
[
  {
    "emphysema": 0.46
  },
  {
    "no_findings": 0.54
  }
]
```

Kód 6 Formát JSONu

výsledek vyplněno hodnotou „None“. Forma předávaného JSONu bude předem domluvena, avšak měla by mít přibližnou podobu viz Kód 6 Formát JSONu.

3.5 Popis implementace

Rozhraní bude implementováno pomocí jazyku Python a frameworku Flask . Důvodem, proč byl vybrán tento framework bylo hned několik. Prvním byla snadná implementace tohoto frameworku, jeho lehkost ve smyslu využití opravdu pouze toho, co je naimportováno, nejedná se o robustní framework. Neméně důležitým prvkem výběru byl požadavek na zabezpečení ze strany zadavatele. Kdy není kladen důraz na role, odlišení uživatelů atp. Pro databázi byl zvolen MySQL systém řízení báze dat. MySQL byl vybrán z důvodu plynulé integrace a neméně důležitým faktorem pro výběr byl fakt, že není potřeba vytvářet složité procedury triggery či jiné funkce, pro které je vhodné použít jiné systémy řízení báze dat. V tomto případě slouží databáze pouze jako úložiště dat bez druhotných zpracování či transformace těchto dat.

Aplikace je zabezpečena pomocí přihlašovacích údajů (hash), které jsou uloženy na serveru jako proměnné prostředí, případně jsou uloženy též v databázi. Přihlašování je vyřešeno tokenem, který uživatel zadá a následně je pomocí něj autentifikován a autorizován. Jedná se o slabší formu zabezpečení, původně však nemělo být zabezpečení řešeno vůbec, tedy toto je minimální ochrana proti vniku neoprávněných osob do systému. Předpokladem je též, že pokud bude aplikace nasazena do provozu, bude provozována na interní síti a přístup z internetu nebude možný.

Aplikace obsahuje drobné optimalizace, které i v dnešní době postrádá značná část aplikací. Jednou z takových optimalizací je stránkování při velkém počtu záznamů. Další optimalizací je načítání pouze části stránky pomocí AJAX volání. Není tak nutné aktualizovat celou stránku. Během dotazu AJAX je zamezeno uživateli kliknout do části stránky, která je aktuálně aktualizována. Důvodem je nemožnost odeslat více požadavků na server a následný problém při zpracování požadavku na straně aplikace a několikanásobné překreslení dané části stránky. Tato optimalizace byla využita pro načítání výsledků, kdy uživatel prochází jednotlivé stránky. Důvodem bylo také zpříjemnění používání systému a přesun mezi stránkami v dolní části detailu pacienta.

Celý projekt je strukturován dle konvencí pro Flask a Python, které jsou uvedeny v jejich oficiální dokumentaci [3]. Pro komunikaci s databází byla použita nástavba SQLAlchemy, která je dostupná v kombinaci s Flask. SQLAlchemy nabízí určitou volnost v dotazování do databáze, kdy vytváří rozhraní mezi přímým dotazování a dotazováním pomocí předem definovaných metod. Tyto metody značně usnadňují práci s danými dotazy.

Pro uvedení do kontextu není nutné vytvářet dotazy přímo v SQL, SQLAlchemy umožňuje využití metod, které následně přetransformuje do SQL za nás. Pro zjednodušení a dodržení konvence DRY (don‘t repeat yourself) byla pro dotazování vytvořena základní třída, která shlukuje znovu použitelné metody pro všechny třídy. Důvodem, proč byla třída vytvořena, je fakt, že s využitím metod z SQLAlchemy, je nutné dbát na to, na jakém objektu provoláváme dané metody. SQLAlchemy zjistí do jaké třídy daný objekt spadá a nad touto entitou provede dotaz. Základní třída obsahuje metody, do kterých pošleme nejen parametr, ale též třídu, nad kterou se má dotaz provést. Metody obsažené v základní třídě obsahují pouze obecné dotazy, tedy update, create, get by id atp, jelikož ostatní metody pro dotazování jsou příliš specifické pro danou entitu, a tak jsou uvedeny ve třídě Manager, který je vytvořen pro každou entitu.

Struktura je rozčleněna do Python balíčků, které v sobě svým názvem zrcadlí svůj obsah. Tedy například pacient bude oddělen od nemoci, nemoc od symptomů atp. Každý z těchto balíčků obsahuje controller. Controller se stará o přijímání požadavků z webových stránek a následné vrácení dat s názvem html stránky, na kterou budou data vykreslena. Dále jsou zde třídy Facade a Manager. Facade je svým způsobem prostředník mezi controllerem a Managerem, dochází zde ke zpracování dat, transformace dat atp. Probíhá zde například dotaz na NS či ES. Pokud je nutné transformovat data, která jsou vrácena z databáze, transformace je provedena v této třídě. Manager se stará o přístup do databáze pomocí metod z SQLAlchemy či předka, ze kterého dané metody dědí, nedochází zde k žádnému jinému zpracování dat. Třída Forms zaštiťuje formuláře, které jsou vytvořeny přímo v aplikaci a odeslány na frontend bez nutnosti složitého vytváření formuláře na frontendu, dochází zde i k předvyplnění dat pomocí k tomu určených metod. Dále je v balíčku zastoupena třída Model, kde je umístěna entita (pacient, nemoc, symptom, atd), která odpovídá databázové tabulce, obsahuje veškeré proměnné a jejich názvy, název tabulky. Poslední třídou umístěnou v balíčku je Utils. Tato třída je pomocná a je spíše určena pro metody, které jsou využívány napříč balíčkem, abychom dodrželi konvenci DRY. Každý balíček má identickou strukturu.

Pro snadnější manipulaci s přesměrováním a vracením jména view (stránka, html soubor obsahující stránku zobrazenou uživateli), byl vytvořen script, který shlukuje názvy view a zpřehledňuje tak práci s těmito názvy, kdy stačí využít název a je vrácen celý název html stránky. Tím je znemožněno udělat překlep v názvu vraceného view. V hlavním kořenovém Python balíčku jsou umístěny další podsložky, které obsahují statické soubory,

konfiguraci a přepis základních formulářových koster v případě nutnosti, kdy potřebujeme upravit základní chování daného formuláře.

Spojení všech Python balíčků probíhá s využitím blueprintů, které nám umožňují nadefinovat nové dekorátory viz Kód 7 Blueprint: Blueprinty umožňují rozřazení projektu jakožto Python balíčku do dalších menších balíčků právě pro uživatele, nemoc atd. Tyto balíčky je však nutné následně zaregistrovat, jinak by aplikace tyto balíčky nebrala v potaz a nebylo by možné tak pracovat s metodami, které jsou v nich specifikované. Dalším důvodem, proč využít blueprinty, je cirkulární import. Cirkulární import znamená, že při importování do jednotlivých python scriptů dochází k zacyklenému importu a aplikace odmítne vykonání a skončí neočekávanou chybou.

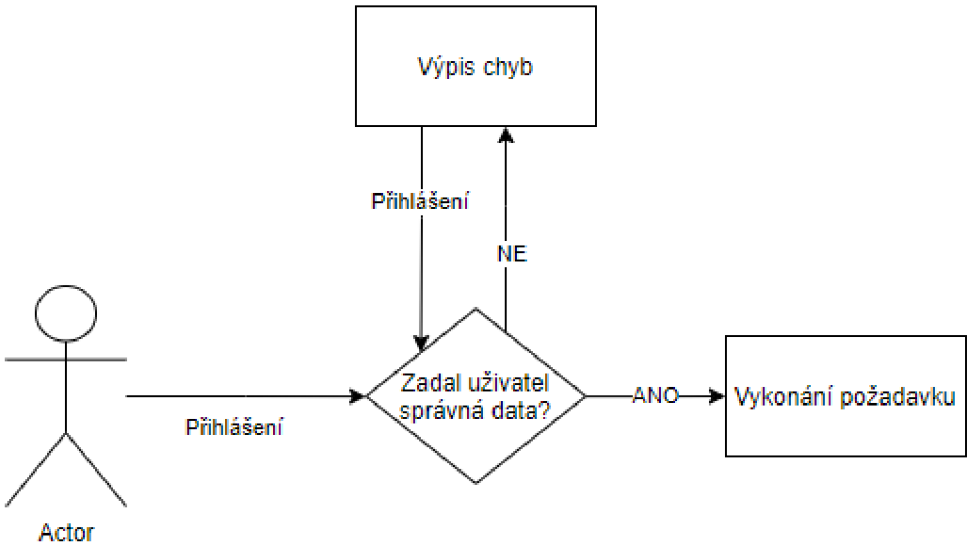
```
@usersBp.route('/login') namísto @app.route('/login')
```

Kód 7 Blueprint

Výsledná aplikace je nasazena na serveru poskytovatele Heroku. Důvodem je užití zdarma a dostupnost. Omezení spočívají v delší odezvě při prvním spuštění. Prvním spuštěním se myslí dotaz, který je od posledního dotazu proveden za více než 30 minut. Dalším omezením je smazání nahraných souborů po dříve zmíněném 30minutovém intervalu. Právě tento interval je nastaven pro vypnutí dynos, které zpracovávají požadavky v Heroku. Heroku bylo zvoleno v závislosti po předchozí zkušenosti.

3.6 Scénáře aplikace

Veškeré operace v aplikaci mají společné pravidlo. Uživatel musí být přihlášený, jinak nedojde ke zpracování požadavku. V opačném případě bude uživatel přesměrován na přihlašovací stránku. Pokud uživatel bude přihlášený, může využívat veškeré funkce aplikace. Obrázek 7 znázorňuje přihlášení uživatele, které je ověřeno před každým požadavkem do aplikace. Jednotlivé scénáře obsahují nákres schématu, jak daná činnost probíhá.

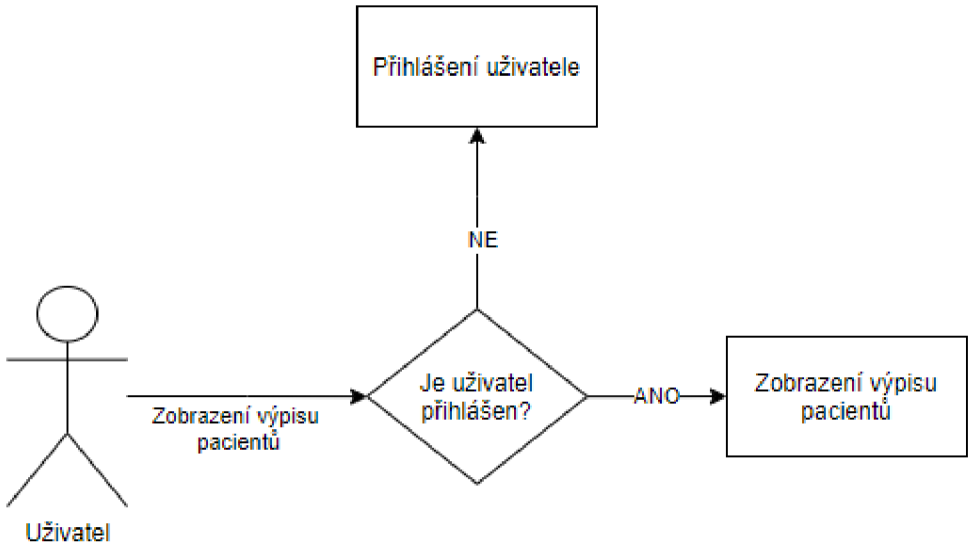


Obrázek 7 Přihlášení uživatele

3.6.1 Operace zobrazení

3.6.1.1 Zobrazení výpisu pacientů

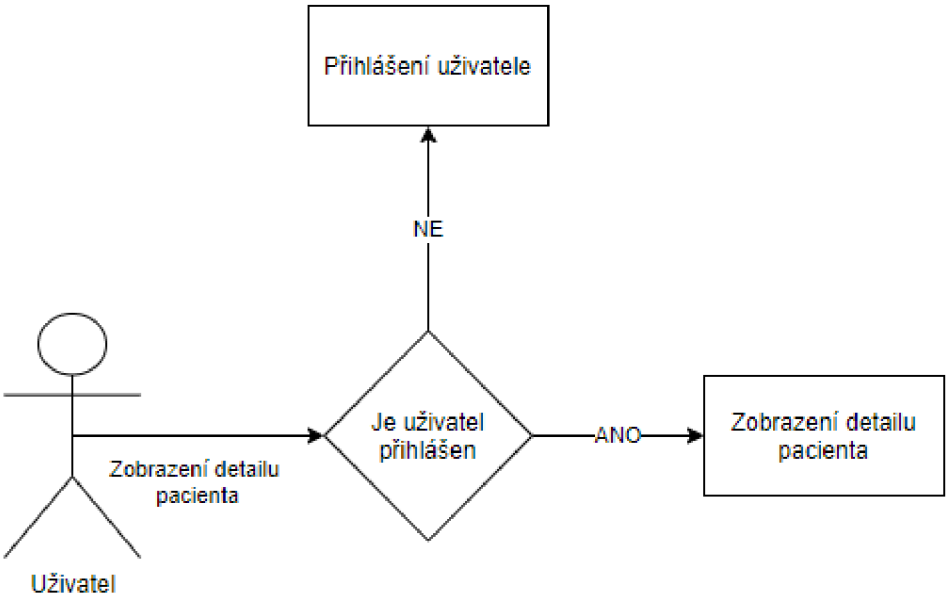
Výpis pacientů je uživateli zobrazen automaticky ihned po přihlášení. Pokud uživatel chce přejít na přehled pacientů, musí zvolit tlačítko s odkazem na výpis pacientů. Po zvolení systém ověří, že je uživatel stále přihlášen. Pokud uživatel je přihlášen, přesměruje ho systém na výpis pacientů. Pokud není přihlášen, je uživatel přesměrován na přihlašovací stránku a opět vyzván k přihlášení.



Obrázek 8 Schéma zobrazení pacientů

3.6.1.2 Zobrazení detailu pacienta

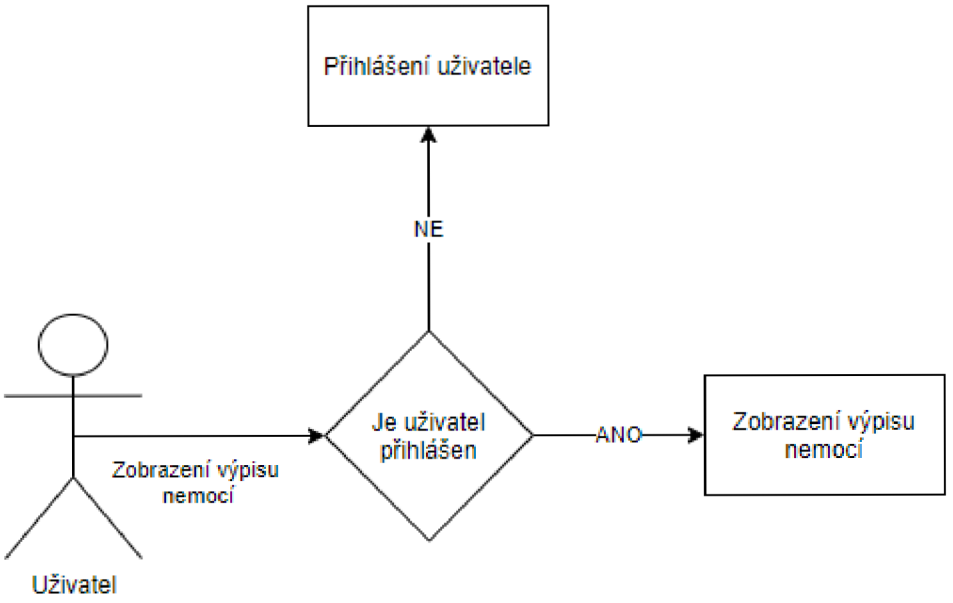
Pokud bude chtít uživatel zobrazit detail pacienta, je předpokladem, že se nachází na výpisu pacientů. Nenachází-li se na výpisu pacientů musí uživatel přejít na scénář Zobrazení výpisu pacientů. Uživatel klikne na symbol „***i***“ na výpisu pacientů u pacienta, pro kterého chce zobrazit detail. Systém zpracuje požadavek a přesměruje uživatele na detail konkrétního pacienta.



Obrázek 9 Schéma zobrazení detailu pacienta

3.6.1.3 Zobrazení výpisu nemocí

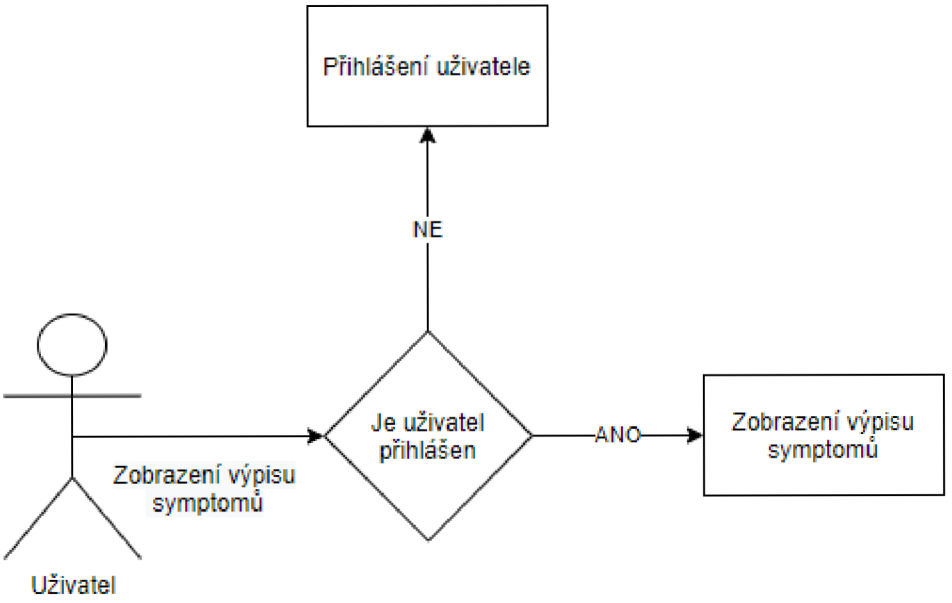
Pro zobrazení výpisu nemocí uživatel klikne na možnost „Nemoci“ v horním navigačním menu. Systém zpracuje požadavek pacienta a přesměruje ho na výpis nemocí.



Obrázek 10 Schéma zobrazení nemocí

3.6.1.4 Zobrazení výpisu symptomů

Pro zobrazení výpisu symptomů uživatel klikne na možnost „Symptomy“ v horním navigačním menu. Systém zpracuje požadavek pacienta a přesměruje ho na výpis symptomů.

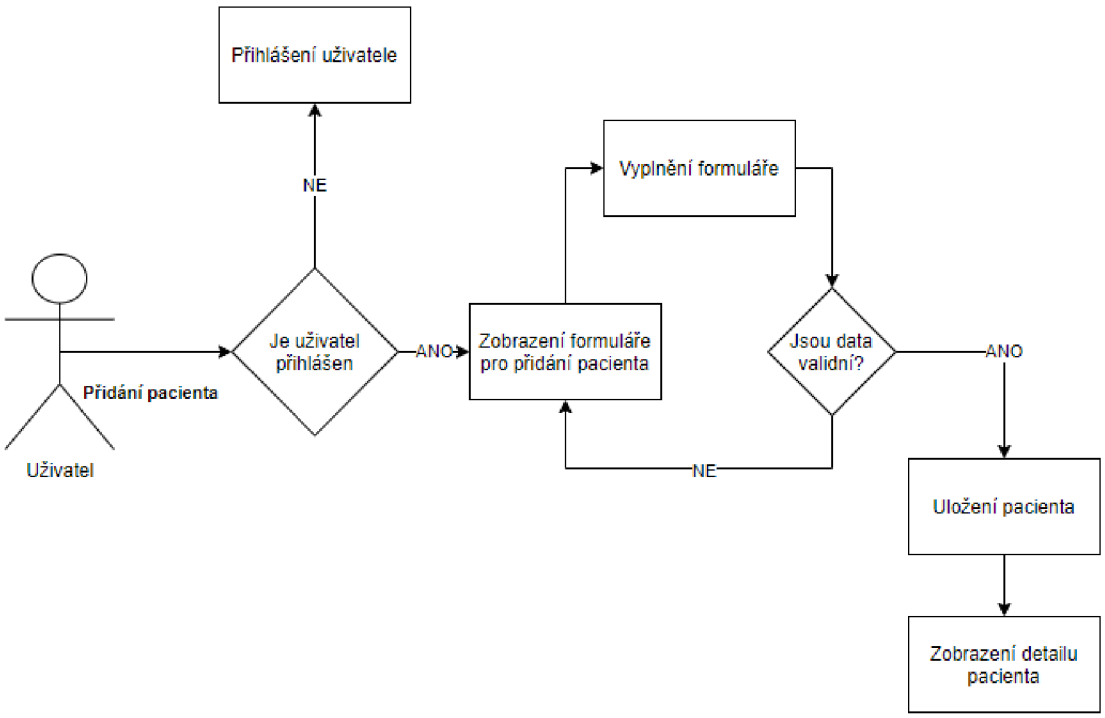


Obrázek 11 Schéma zobrazení symptomů

3.6.2 Operace create/přidat

3.6.2.1 Přidání pacienta

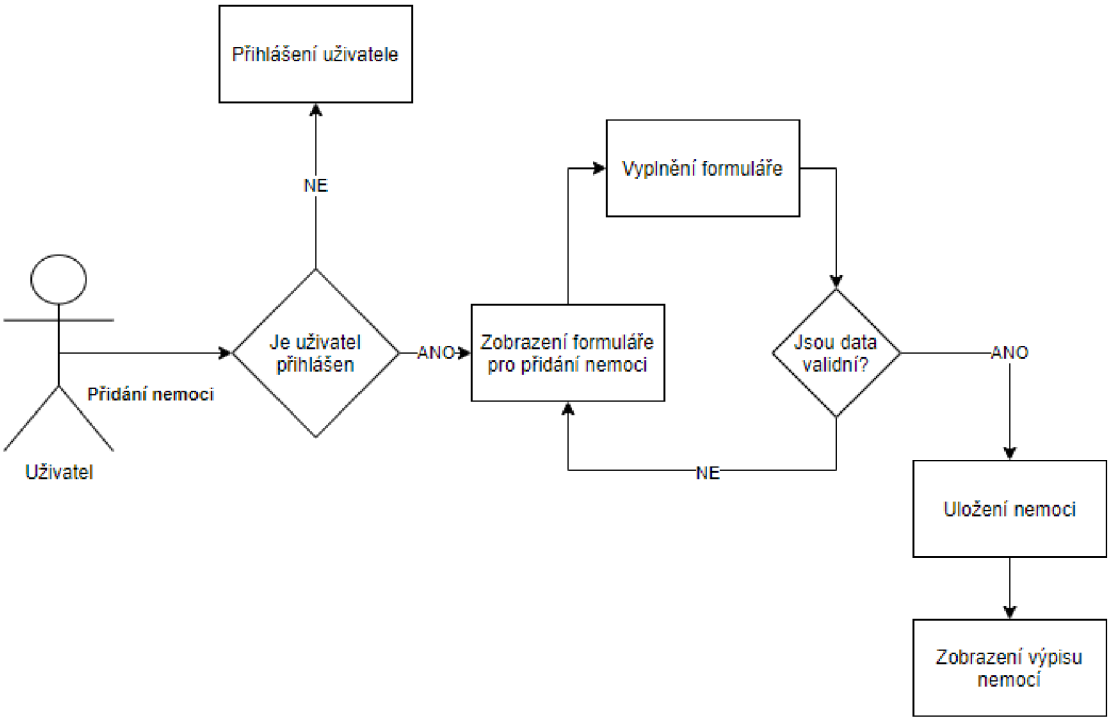
Pro přidání pacienta do systému bude muset uživatel přejít na hlavní stránku pacientů viz Zobrazení výpisu pacientů. Na této stránce je umístěno tlačítko „Registrace nového pacienta“. Po kliknutí na tlačítko bude uživatel přesměrován na formulář pro registraci nového pacienta. Formulář obsahuje povinná a nepovinná data, která jsou následně využívána v ES. Nepovinná data jsou specifická, ne každý lékař má přístup k těmto informacím. Tyto položky jsou na první pohled schovány pod možností „Zobrazit pokročilé nastavení“ Uživatel tyto hodnoty vyplní a klikne na tlačítko „Registrace pacienta“. Pokud uživatel vyplnil hodnoty správně, bude přesměrován na detail vytvořeného záznamu o pacientovi. Pokud nevyplnil hodnoty správně, bude přesměrován zpět na formulář, kde pod chybnými inputy formuláře bude uvedena chyba, které se uživatel dopustil, respektive bude zde vysvětleno, co daný input přijímá. Tedy pokud uživatel vloží nevalidní hodnotu do pole pro číslo, bude vrácena chyba, která uživateli sdělí, že musí zadat číselné hodnoty. Po napravení chyb může uživatel opět odeslat záznam pacienta k uložení.



Obrázek 12 Schéma přidání pacienta

3.6.2.2 Přidání nemoci

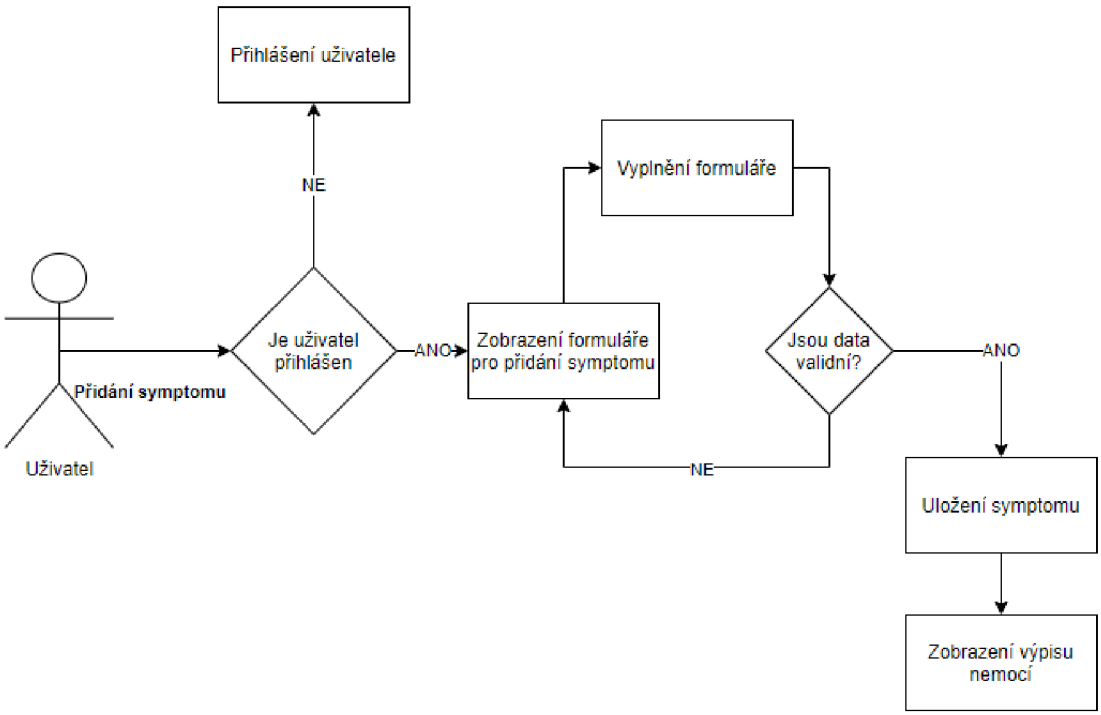
Přidání nemoci probíhá přes formulář, který slouží pouze pro zadání názvu nemoci. Formulář je dosažitelný ze záložky „Nemoci“. Po kliknutí na tlačítko „Vytvoření nemoci“ bude uživatel přesměrován na formulář pro vytvoření záznamu o nemoci. Uživatel hodnotu vyplní a klikne na tlačítko „Uložit“. Pokud uživatel vyplnil hodnoty správně, bude přesměrován na výpis nemocí. Pokud nevyplnil hodnoty správně, bude přesměrován zpět na formulář, kde bude pod inputem uvedena chyba, které se uživatel dopustil. Po opravení chyb může uživatel opět odeslat záznam nemoci k uložení. Tato funkcionalita spolu s přidáním symptomů je vytvářena dynamicky za běhu systému pro budoucí možnost využití tohoto systému, nejen pro identifikaci plicního emfyzému.



Obrázek 13 Schéma přidání nemoci

3.6.2.3 Přidání symptomu

Přidání symptomu se vždy váže k vytvořené nemoci, nelze tedy vytvořit symptom bez nemoci. Předpokladem pro přidání symptomu je, že se uživatel nachází na záložce s výpisem nemocí. Na této stránce je vypsán seznam nemocí a u každé nemoci je odkaz „Přidat symptom“. Po prokliknutí se uživateli zobrazí formulář s jedním inputem, kam je možné zadat název symptomu. Pod tímto inputem se nachází selectbox, který umožňuje přidat symptom nejen k nemoci, přes kterou jsme se dostali na vytváření symptomů, ale také pro jiné nemoci. Důvodem vytvoření této funkcionality bylo povědomí o stejných symptomech pro různé nemoci. Uživatel vyplní tento input a vybere pro jaké nemoci chce, aby se symptom uložil. Systém validuje vstup. Pokud je vstup správný, uloží pro každou nemoc jeden záznam se symptomem, pokud je vstup chybný, je uživatel přesměrován zpět na formulář založení a vyzván k nápravě.

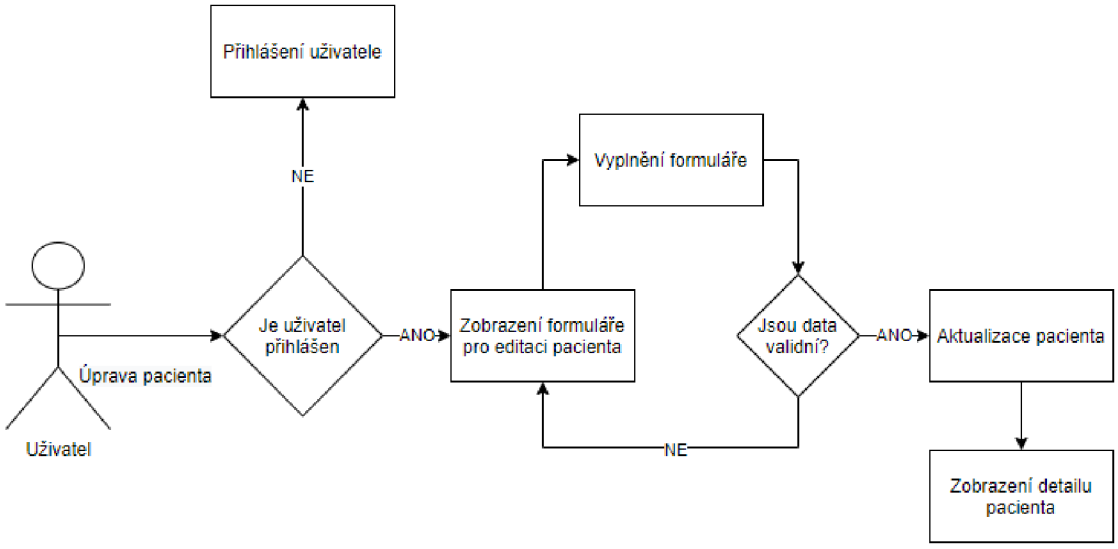


Obrázek 14 Schéma uložení symptomu

3.6.3 Operace update/upravit

3.6.3.1 Úprava záznamu o pacientovi

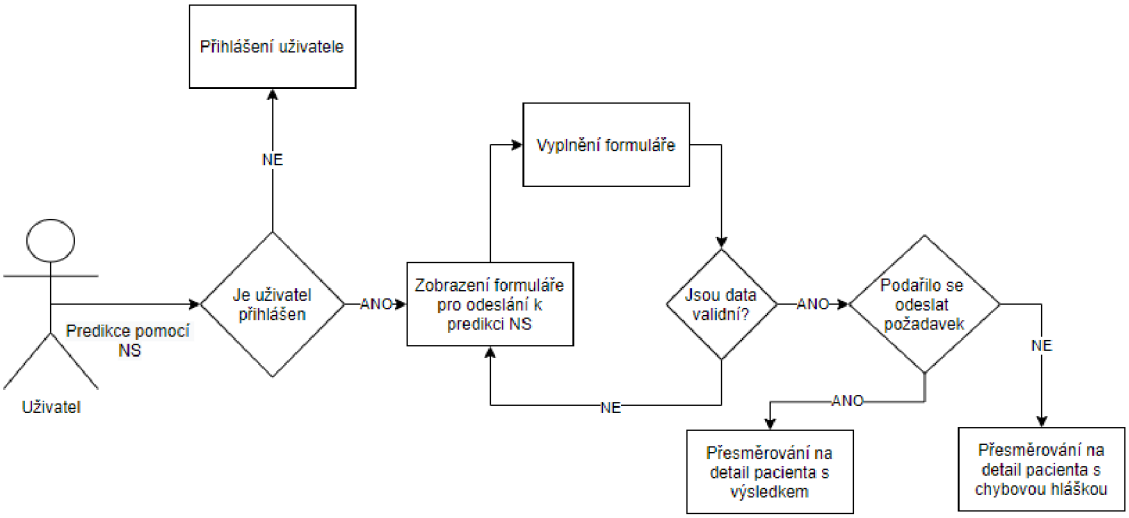
Úprava záznamu o pacientovi je přístupná přes výpis pacientů. Na tomto výpisu je pro každý záznam pacienta přístupná ikona tužky. Po prokliknutí přes tuto ikonu je uživatel přesměrován na formulář identický s formulářem pro tvorbu záznamu o pacientovi. Tentokrát budou však data předvyplněna hodnotami, které záznam o pacientovi disponuje a jsou uloženy v databázi. Uživatel edituje požadované vlastnosti a kliká na tlačítko „Aktualizovat pacienta“. Systém validuje vstup. Pokud je vstup validní aktualizuje pacienta, pokud není validní, vypíše uživateli hlášky s chybami.



Obrázek 15 Schéma úpravy pacienta

3.6.4 Operace odeslání do predikce NS

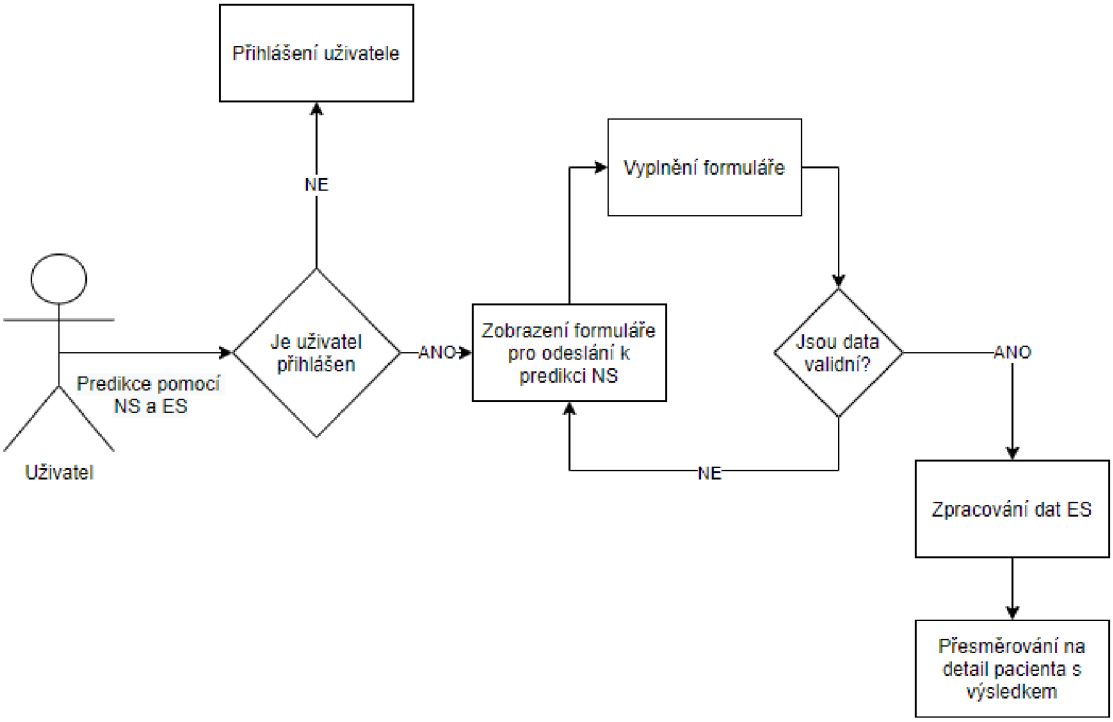
Uživateli je umožněno odeslat pouze snímek RTG do neuronové sítě bez toho, aby se daný výsledek zpracoval v expertním systému. Uživatel musí mít zobrazenou stránku s daným pacientem. Pod informacemi o pacientovi se nachází tlačítko „Vytvořit predikci pomocí NS“. Uživatel je přesměrován na stránku, kde musí vložit snímek RTG, a to ve formátu JPG anebo PNG. Po vložení uživatel klikne na tlačítko „Odeslat k predikci“. Systém validuje vstup, pokud je zde obrázek jiného než dovoleného vstupu, je přesměrován zpět, kde je mu umožněno vybrat snímek ve správném formátu. Systém se pokusí odeslat snímek do neuronové sítě. Pokud požadavek selže, je uživatel na detailu pacienta informován o selhání jeho požadavku. Pokud je požadavek zpracován, dojde k přesměrování uživatele na detail pacienta s vyplněným výsledkem z NS. Důvodem vytvoření tohoto scénáře je zajištění zpracování snímku RTG i bez dat, jelikož každý lékař nemusí mít přístup k potřebným informacím o pacientovi.



Obrázek 16 Schéma odeslání k predikci NS

3.6.5 Operace odeslání do predikce NS+ES

Plnohodnotnou funkcionalitou je odeslání snímku do NS a dat o pacientovi do ES, přičemž do ES je navíc odeslán výsledek z NS a validuje se jeho správnost. Uživatel musí mít zobrazenou stránku s daným pacientem. Pod informacemi o pacientovi se nachází tlačítko „Vytvořit predikci pomocí NS a expertního systému“. Uživatel je přesměrován na stránku, kde může vložit snímek RTG, a to ve formátu JPG anebo PNG. Dále se zde nachází formulář s předvyplněnými daty o pacientovi. Tato data lze před odesláním upravit. Po vložení/upravení uživatel klikne na tlačítko „Odeslat k predikci“. Systém validuje vstup, pokud se nachází ve formuláři chyba, je uživatel přesměrován zpět, aby hodnoty upravil dle povolených hodnot. Systém se pokusí odeslat snímek do neuronové sítě. Pokud požadavek selže, je uživatel na detailu pacienta informován o selhání jeho požadavku. Pokud uspěje, je výsledek zpracován a odeslán spolu s ostatními daty do expertního systému. Expertní systém prochází data o pacientovi a výsledek z NS. Po dokončení úlohy je uživatel přesměrován na detail pacienta, kde se nachází výsledek z expertního systému a NS. Aby bylo možné systém využívat, je prozatím povoleno odesílat v tomto scénáři data bez snímku RTG. Po napojení na NS bude toto chování zakázáno.



Obrázek 17 Schéma odeslání k predikci NS + ES

4 Výsledky a shrnutí konzultace s lékařem

Návrh systému inicioval doménový expert (lékař), s ním byla následně aplikace konzultována. V závěru prací expert pomohl dodefinovat dodatečná pravidla pro ES. Pravidla a zároveň parametry těchto pravidel napomáhají určit, zdali výsledek z neuronové sítě je pravdivý či nikoliv, respektive, zdali se na tento výsledek dá spolehnout při diagnostikování plicního emfyzému. ES docílí těchto výsledků právě po poskytnutí dat z NS, která zpracovává rentgenový snímek.

ES systém může fungovat nezávisle na NS, kdyby například došlo k výpadku mezi ES a NS. Při výpadku NS se nebudou moci pravidla z ES „opřít“ o výsledek poskytnutý z NS, a tak bude finální výsledek pouze hrubé odvození nemoci na základně vstupních parametrů. Některé z parametrů jsou dopočítávány na základě vzorců, které jsou veřejně známé (BMI) anebo na základě konzultace s lékařem (FEV1). Pohled na parametry zkušenějšími lékaři mohou ihned napovědět zastoupení emfyzému, protože se nejedná o značně robustní systém jako byly v minulosti například systémy MYCIN atp. Záměrem práce nebylo však vytvořit takovýto systém. ES může napomoci při rozhodování a dokáže zpřesnit výsledek NS, což bylo účelem tohoto ES.

Jelikož lze však ES systém využít bez NS pouze na základě parametrů, byly otestovány tyto vstupy a následně dle definovaných pravidel korektně určen emfyzém, či pouze výčet problémů, kterými pacient trpí. Výsledek těchto pokusů byl uspokojivý a definovaná pravidla zafungovala korektně.

5 Závěr

Cílem této práce bylo vytvořit aplikaci a ES, která by prezentovala uživateli grafický výstup, bylo by zde možné zjišťovat, zdali uživatel trpí plicním emfyzémem, či nikoliv. Aplikace by měla zajistit přehled pro lékaře/uživatele, kdy budeme udržovat informace o pacientovi včetně provedených tesů do NS a ES+NS.

Vytvoření aplikace pro prezentaci dat bylo úspěšné. Tato aplikace jest dostupná na webové adrese https://emphysema.herokuapp.com/, kdy po zadání hesla: *2a5708b107859e3aeb0021ea986658ffd100f51daf3e058ac42eda1ca5442956* je uživatel vpuštěn do aplikace a přesměrován na přehled pacientů. Byly zde úspěšně implementovány veškeré CRUD operace, které jsou nutné pro chod aplikace tohoto typu. Dodržením PEP specifikací bylo dosaženo poměrně dobře čitelného a znovupoužitelného kódu, který bude v budoucnu možno rozšířit o další výsledky a další neuronové sítě. V takovémto případě však bude nutné částečně aplikaci upravit a zřídit zde asynchronní komunikaci, která uživatelům zpříjemní práci se systémem, jelikož dotazování na více nežli jednu síť je poměrně časově náročné. Časová náročnost se také odvíjí od dostupného výkonu a složitosti predikce.

Momentálně není NS dostupná, a tak ES nemůže zcela korektně pracovat dle nastavených pravidel, která byla s tímto ohledem vytvořena. ES se totiž značně spoléhá na výstup z NS na identifikaci plicního emfyzému. Využití ES dává smysl jako celek v kombinaci s NS. Kompletace pravidel byla však provedena tak, aby bylo možné ES využít též bez NS a síť mohla být napojena či vyměněna na pozadí, na serverové části. Dílčím cílem této práce však bylo ověřit funkčnost kombinace ES a NS. Tento cíl nemohl být splněn z důvodu chybějícího prvku a to NS, která nebyla součástí této práce. Z tohoto důvodu je ES vybaven pravidly, která mu umožňují pracovat též bez NS. Je však nutné opět zdůraznit, že za plně fungující systém můžeme považovat až kombinaci NS a ES.

Implementace pravidel, která pracují bez NS proběhla úspěšně a predikce vždy odpovídaly očekávanému výsledku.

Vzhledem k aktuální situaci ve světě, která je spojena s výskytem nemoci COVID-19, je identifikování nemocí plic důležitější než kdy předtím. Zároveň však tento enormní zájem způsobuje útlum ve výzkumu spojení například ES a NS, který by mohl posunout identifikaci a predikci na další úroveň. Důvodem je validace výstupu z neuronové sítě v závislosti na reálných a aktuálních datech o pacientovi, které zadal lékař a které nejsou

v RTG snímcích bez vnějšího zásahu přítomna. Tento projekt, spojení NS a ES, by mohl v budoucnosti vést k sofistikovanějším výsledkům, které by byly opřené o jasně definovaná fakta.

6 Zkratky

ES = Expertní systém

NS = Neuronová síť

RTG = rentgen

BA = Backpropagation algoritmus

PEP = Python Enhancement Proposals

DRY = do not repeat yourself

FEV = objem vzduchu vydechnutý s největším úsilím za 1. sekundu po maximální nádechu

FVC = maximální objem vzduchu, který lze po maximálním nádechu prudce vydechnout

DLCO = difuzní kapacita plic pro oxid uhelnatý

BMI = index tělesné hmotnosti

CHOPN = chronická obstrukční nemoc

7 Seznam obrázků, tabulek a grafů

8 Seznam použité literatury

1. CELBOVÁ, Iva. Úvod do problematiky expertních systémů. Ikaros [online]. 1999, ročník 3, číslo 8 [cit. 2021-09-28]. urn:nbn:cz:ik-10378. ISSN 1212-5075. Dostupné z: http://ikaros.cz/node/10378
2. MARTÍNEZ PÉREZ, Roberto Abdelkader. Experta. *Read the Docs - Experta* [online]. 2019 [cit. 2021-9-28]. Dostupné z: https://experta.readthedocs.io/en/latest/
3. General Python FAQ¶. *Python* [online]. Python Software Foundation, c2001-2021 [cit. 2021-9-28]. Dostupné z: https://docs.python.org/3/faq/general.html#what-is-python
4. What is WSGI? *WSGI.org* [online]. [cit. 2021-9-28]. Dostupné z: https://wsgi.readthedocs.io/en/latest/what.html
5. LEE, S.Y. CLIPS. *S.Y. Lee* [online]. Taipei, Taiwan [cit. 2021-9-28]. Dostupné z: https://www.csie.ntu.edu.tw/~sylee/courses/clips/bpg/nodePreface.html
6. Dopředná Neuronová síť. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2021-9-28]. Dostupné z: https://commons.wikimedia.org/w/index.php?curid=1496812
7. NIKOLOPOULOS, Chris. *Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems*. New York: Marcel Dekker, 1997. ISBN 0-8247-9927-5.
8. KVASNIČKA, Vladimír. *Úvod do teórie neurónových sietí*. [Slovenská republika]: IRIS, 1997. ISBN 80-887-7830-1.
9. LUCAS, Peter J.F. and Linda C. van der Gaag. "Principles of expert systems." *International computer science series* (1991) [cit. 2021-9-28]. Dostupné z: https://www.researchgate.net/profile/Hazim-Tahir/post/Do_you_use_expert_systems_in_scientific_work/attachment/59d62b3a79197b807798995d/AS%3A342356722241549%401458635611889/download/proe.pdf
10. RAJPURKAR, Pranav, Jeremy IRVIN, Robyn L. BALL, et al. Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists. *PLOS Medicine* [online]. 2018, **15**(11) [cit. 2021-9-28]. ISSN 1549-1676. Dostupné z: doi:10.1371/journal.pmed.1002686

11. DARLINGTON, Keith W. Designing for Explanation in Health Care Applications of Expert Systems. *SAGE Open* [online]. 2011, **1**(1) [cit. 2021-9-28]. ISSN 2158-2440. Dostupné z: doi:10.1177/2158244011408618
12. HOLČÍK, Jiří a Martin KOMENDA. *Matematická biologie: e-learningová učebnice [online].* [online]. 1. Brno: Masarykova univerzita, 2015 [cit. 2021-9-28]. ISBN 978-80-210-8095-9. Dostupné z: https://portal.matematickabiologie.cz/
13. HELLER, Israel, Aharon ISAKOV, Yael VILLA, et al. Evaluation of CaDet, a computer-based clinical decision support system for early cancer detection: a comparison with the performance of clinicians. *Cancer Detection and Prevention* [online]. 2004, **28**(5), 352-356 [cit. 2021-9-28]. ISSN 0361090X. Dostupné z: doi:10.1016/j.cdp.2004.06.004
14. DEVAN, K.S., P.A. VENKATACHALAM a A.F.M. HANI. Expert system with an embedded imaging module for diagnosing lung diseases. In: *Proceedings of 7th International Workshop on Enterprise networking and Computing in Healthcare Industry, 2005. HEALTHCOM 2005* [online]. IEEE, 2005, s. 229-234 [cit. 2021-9-28]. ISBN 0-7803-8940-9. Dostupné z: doi:10.1109/HEALTH.2005.1500445
15. BASSEM S, Abu-Nasser. Medical Expert Systems Survey. *International Journal of Engineering and Information Systems* [online]. 2017, **1**(7), 218-224 [cit. 2021-9-30]. ISSN 2000-000X. Dostupné z: https://hal.archives-ouvertes.fr/hal-01610722/document
16. SON, Le Hoang, Tran Manh TUAN, Hamido FUJITA, Nilanjan DEY, Amira S. ASHOUR, Vo Truong Nhu NGOC, Le Quynh ANH a Dinh-Toi CHU. Dental diagnosis from X-Ray images: An expert system based on fuzzy computing. *Biomedical Signal Processing and Control* [online]. 2018, **39**, 64-73 [cit. 2021-9-30]. ISSN 17468094. Dostupné z: doi:10.1016/j.bspc.2017.07.005
17. ABU-NASER, S.S., H. El- HISSI, M. Abu- RASS a N. El- KHOZONDAR. An Expert System for Endocrine Diagnosis and Treatments using JESS. *Journal of Artificial Intelligence* [online]. 2010, **3**(4), 239-251 [cit. 2021-9-30]. ISSN 19945450. Dostupné z: doi:10.3923/jai.2010.239.251
18. MANIKANDAN, R., Ambeshwar KUMAR a Deepak GUPTA. Hybrid computational intelligence for healthcare and disease diagnosis. *Hybrid Computational Intelligence* [online]. Elsevier, 2020, 2020, s. 97-122 [cit. 2021-9-

30]. ISBN 9780128186992. Dostupné z: doi:10.1016/B978-0-12-818699-2.00006-8

19. KARIM, Razuan, Karl ANDERSSON, Mohammad Shahadat HOSSAIN, Md. Jasim UDDIN a Md. Perveg MEAH. A belief rule based expert system to assess clinical bronchopneumonia suspicion. In: *2016 Future Technologies Conference (FTC)* [online]. IEEE, 2016, 2016, s. 655-660 [cit. 2021-9-30]. ISBN 978-1-5090-4171-8. Dostupné z: doi:10.1109/FTC.2016.7821675
20. BEHZADI-KHORMOUJI, Hamed, Habib ROSTAMI, Sana SALEHI, et al. Deep learning, reusable and problem-based architectures for detection of consolidation on chest X-ray images. *Computer Methods and Programs in Biomedicine* [online]. 2020, **185** [cit. 2021-9-30]. ISSN 01692607. Dostupné z: doi:10.1016/j.cmpb.2019.105162
21. TAYLOR, Andrew G., Clinton MIELKE, John MONGAN a Suchi SARIA. Automated detection of moderate and large pneumothorax on frontal chest X-rays using deep convolutional neural networks: A retrospective study. *PLOS Medicine* [online]. 2018, **15**(11) [cit. 2021-9-30]. ISSN 1549-1676. Dostupné z: doi:10.1371/journal.pmed.1002697
22. IGNIZIO, James P. A brief introduction to expert systems. *Computers & Operations Research* [online]. 1990, **17**(6), 523-533 [cit. 2021-9-30]. ISSN 03050548. Dostupné z: doi:10.1016/0305-0548(90)90058-F
23. SALEM, Hesham, Daniele SORIA, Jonathan N. LUND a Amir AWWAD. A systematic review of the applications of Expert Systems (ES) and machine learning (ML) in clinical urology. *BMC Medical Informatics and Decision Making* [online]. 2021, **21**(1) [cit. 2021-10-8]. ISSN 1472-6947. Dostupné z: doi:10.1186/s12911-021-01585-9
24. S. TODD, Brian. *An Introduction to Expert Systems*. 95. Oxford: Oxford University Computing Laboratory, Programming Research Group, 1992. ISBN 0902928732.
25. *Funkční vyšetření kardiorespiračního systému* [online]. c2021 [citováno 31. 10. 2021]. Dostupný z: <https://www.wikiskripta.eu/index.php?title=Funk%C4%8Dn%C3%AD_vy%C5%A1e

t%C5%99en%C3%AD_kardiorespira%C4%8Dn%C3%ADho_syst%C3%A9mu&oldid=448580>

26. STOIA, Claudiu-Leonardo. A Study Regarding the Use of Expert Systems in Economics Field. *Procedia Economics and Finance* [online]. 2013, **6**, 385-391 [cit. 2021-11-04]. ISSN 22125671. Dostupné z: doi:10.1016/S2212-5671(13)00152-4
27. HEILMAN, James. Emphysema2008. *Wikipedia* [online]. 2010 [cit. 2021-11-04]. Dostupné z: https://commons.wikimedia.org
28. L. SNIDER, Gordon, Jerome KLEINERMAN, William M. THURLBECK a Zakir H. BENGALI. The Definition of Emphysema: Report of a National Heart, Lung, and Blood Institute, Division of Lung Diseases Workshop. *American Review of Respiratory Disease* [online]. 1985, **132**(1), 182–185 [cit. 2021-11-04]. Dostupné z: doi:10.1164/arrd.1985.132.1.182
29. BADNJEVIC, Almir, Lejla GURBETA a Eddie CUSTOVIC. An Expert Diagnostic System to Automatically Identify Asthma and Chronic Obstructive Pulmonary Disease in Clinical Settings. *Scientific Reports* [online]. 2018, **8**(1) [cit. 2021-11-05]. ISSN 2045-2322. Dostupné z: doi:10.1038/s41598-018-30116-2
30. BRAIDO, Fulvio, Pierachille SANTUS, Angelo CORSICO, Fabiano DI MARCO, Giovanni MELIOLI, Nicola SCICHILONE a Paolo SOLIDORO. Chronic obstructive lung disease “expert system&rdquo: validation of a predictive tool for assisting diagnosis. *International Journal of Chronic Obstructive Pulmonary Disease* [online]. 2018, **13**, 1747-1753 [cit. 2021-11-05]. ISSN 1178-2005. Dostupné z: doi:10.2147/COPD.S165533
31. ISMAEL, Aras M. a Abdulkadir ŞENGÜR. Deep learning approaches for COVID-19 detection based on chest X-ray images. *Expert Systems with Applications* [online]. 2021, **164** [cit. 2021-11-05]. ISSN 09574174. Dostupné z: doi:10.1016/j.eswa.2020.114054
32. OZTURK, Tulin, Muhammed TALO, Eylul Azra YILDIRIM, Ulas Baran BALOGLU, Ozal YILDIRIM a U. RAJENDRA ACHARYA. Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Computers in Biology*

and Medicine [online]. 2020, **121** [cit. 2021-11-05]. ISSN 00104825. Dostupné z: doi:10.1016/j.compbiomed.2020.103792

33. NARIN, Ali, Ceren KAYA a Ziynet PAMUK. Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks. *Pattern Analysis and Applications* [online]. 2021, **24**(3), 1207-1220 [cit. 2021-11-05]. ISSN 1433-7541. Dostupné z: doi:10.1007/s10044-021-00984-y
34. SOUZA, Johnatan Carvalho, João Otávio BANDEIRA DINIZ, Jonnison Lima FERREIRA, Giovanni Lucca FRANÇA DA SILVA, Aristófanes CORRÊA SILVA a Anselmo Cardoso DE PAIVA. An automatic method for lung segmentation and reconstruction in chest X-ray using deep neural networks. *Computer Methods and Programs in Biomedicine* [online]. 2019, **177**, 285-296 [cit. 2021-11-05]. ISSN 01692607. Dostupné z: doi:10.1016/j.cmpb.2019.06.005
35. AHMED, M. A., Z.T. AL-QAYSI, Moceheb Lazam SHUWANDY, Mahmood Maher SALIH a Majid Hamid ALI. Automatic COVID-19 pneumonia diagnosis from x-ray lung image: A Deep Feature and Machine Learning Solution. *Journal of Physics: Conference Series* [online]. 2021, **1963**(1) [cit. 2021-11-05]. ISSN 1742-6588. Dostupné z: doi:10.1088/1742-6596/1963/1/012099
36. *Welcome to Flask* [online]. Pallets, c2010 [cit. 2021-11-05]. Dostupné z: https://flask.palletsprojects.com/en/2.0.x/
37. *Django* [online]. Django Software Foundation, 2021 [cit. 2021-11-05]. Dostupné z: https://www.djangoproject.com/
38. GUARRASI, Valerio, Natascha Claudia D'AMICO, Rosa SICILIA, Ermanno CORDELLI a Paolo SODA. A Multi-Expert System to Detect COVID-19 Cases in X-ray Images. In: *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)* [online]. IEEE, 2021, 2021, s. 395-400 [cit. 2021-11-05]. ISBN 978-1-6654-4121-6. Dostupné z: doi:10.1109/CBMS52027.2021.00090

9 Přílohy

9.1 Báze pravidel

```
class PredictIllness(KnowledgeEngine):
    result = ''

    @DefFacts()
    def init_emphysema_facts(self):
        yield Emphysema(smoker=True)
        yield PtfFev1VerySevere(maxValue=49, minValue=35)
        yield PtfFev1Severe(maxValue=59, minValue=50)
        yield PtfFev1ModeratelySevere(maxValue=69, minValue=60)
        yield PtfFev1Moderate()
        yield PtfFev1Mild()
        yield PtfDlcoVerySevere(maxValue=39, minValue=0)
        yield PtfDlcoModerate(maxValue=59, minValue=40)
        yield PtfDlcoMild(maxValue=79, minValue=60)
        yield BodyMassIndex(underweight=18.5, normal_weight_min=18.5,
normal_weight_max=24.9, overweight_min=25.0,
                            overweight_max=29.9, obesity=30.0)
        yield PulseOxymetry(minPulseOxymetry=95)

    @Rule(Symptoms(val=MATCH.symptoms),
          SymptomsToDisease(num=MATCH.all_symptoms_for_disease,
disease=MATCH.disease),
          TEST(lambda symptoms, disease: increment(symptoms, disease)))
    def detect_disease_0(self, symptoms, all_symptoms_for_disease,
disease):
        if len(symptoms) == all_symptoms_for_disease:
            self.result = disease
            self.declare(Prediction('matches_all_symptoms'))

    @Rule(AND(Prediction('matches_all_symptoms'),
              NOT(Prediction('FEV1_decreased'))),
          PtfFev1FromPatient(MATCH.FEV1),
          TEST(lambda FEV1: FEV1 is not None),
          PtfFev1VerySevere(maxValue=MATCH.FEV1VSMAX,
minValue=MATCH.FEV1VSMIN),
          PtfFev1Severe(maxValue=MATCH.FEV1SMAX,
minValue=MATCH.FEV1SMIN),
          OR(AND(TEST(lambda FEV1, FEV1VSMAX: FEV1 <= FEV1VSMAX),
                 TEST(lambda FEV1, FEV1VSMIN: FEV1 >= FEV1VSMIN)),
             AND(TEST(lambda FEV1, FEV1SMAX: FEV1 <= FEV1SMAX),
                 TEST(lambda FEV1, FEV1SMIN: FEV1 >= FEV1SMIN))))
    def detect_disease_1(self, FEV1):
        self.result += "\r\n Velmi nízké hodnoty FEV1 " + str(FEV1)
        self.declare(Prediction('FEV1_decreased'))

    @Rule(NOT(Prediction('FEV1_decreased')),
          PtfFev1FromPatient(MATCH.FEV1),
          TEST(lambda FEV1: FEV1 is not None),
```

```
            PtfFev1VerySevere(maxValue=MATCH.FEV1VSMAX,
minValue=MATCH.FEV1VSMIN),
            PtfFev1Severe(maxValue=MATCH.FEV1SMAX,
minValue=MATCH.FEV1SMIN),
            OR(AND(TEST(lambda FEV1, FEV1VSMAX: FEV1 <= FEV1VSMAX),
                   TEST(lambda FEV1, FEV1VSMIN: FEV1 >= FEV1VSMIN)),
               AND(TEST(lambda FEV1, FEV1SMAX: FEV1 <= FEV1SMAX),
                   TEST(lambda FEV1, FEV1SMIN: FEV1 >= FEV1SMIN))))
    def detect_disease_2(self, FEV1):
        self.result += "\r\n Velmi nízké hodnoty FEV1 " + str(FEV1)
        self.declare(Prediction('FEV1_decreased'))

    @Rule(AND(NOT(Prediction('matches_all_symptoms')),
NOT(Prediction('LOW_FEV1_DLCO')), Prediction('FEV1_decreased'),
              Prediction('DLCO_decreased')),
            PtfDlcoFromPatient(MATCH.DLCO),
            PtfFev1FromPatient(MATCH.FEV1))
    def detect_disease_3(self, FEV1, DLCO):
        self.result = "\r\n Velmi nízké hodnoty FEV1 {} a DLCO {} \r\n ->
indikátor Emfyzému".format(FEV1, DLCO)
        self.declare(Prediction('LOW_FEV1_DLCO'))

    @Rule(AND(Prediction('matches_all_symptoms'),
Prediction('FEV1_decreased')),
            PtfDlcoFromPatient(MATCH.DLCO),
            TEST(lambda DLCO: DLCO is not None),
            PtfDlcoVerySevere(maxValue=MATCH.DLCOVSMAX,
minValue=MATCH.DLCOVSMIN),
            AND(TEST(lambda DLCO, DLCOVSMAX: DLCO <= DLCOVSMAX),
                TEST(lambda DLCO, FEV1VSMIN: DLCO >= FEV1VSMIN)))
    def detect_disease_4(self, DLCO):
        self.result += "\r\n Velmi nízké hodnoty DLCO " + str(DLCO)
        self.declare(Prediction('DLCO_decreased'))

    @Rule(AND(Prediction('matches_all_symptoms'),
Prediction('LOW_FEV1_DLCO')),
            PulseOxymetryFromPatient(MATCH.OXYGEN),
            TEST(lambda OXYGEN: OXYGEN is not None),
            PulseOxymetry(minPulseOxymetry=MATCH.minPulseOxymetry),
            TEST(lambda OXYGEN, minPulseOxymetry: OXYGEN <
minPulseOxymetry))
    def detect_disease_5(self, OXYGEN):
        self.result += "\r\n Nižší hodnoty SpO2 " + str(OXYGEN)
        self.declare(Prediction('SpO2_decrease'))

    @Rule(OR(AND(Prediction('matches_all_symptoms'),
Prediction('DLCO_decreased'), Prediction('FEV1_decreased'),
                 Prediction('SpO2_decrease'),
PredictionBMI('bmi_checked'), NOT(Prediction('smoker'))),
             AND(Prediction('DLCO_decreased'),
Prediction('FEV1_decreased'),
```

```
                Prediction('SpO2_decrease'),
NOT(Prediction('smoker')))))
    def detect_disease_6(self):
        self.declare(Prediction('smoker'))

    @Rule(NOT(PredictionBMI('bmi_checked')),
          BodyMassIndexFromPatient(MATCH.BMI),
          TEST(lambda BMI: BMI is not None),
          BodyMassIndex(obesity=MATCH.obesity),
          TEST(lambda obesity, BMI: BMI > obesity))
    def detect_disease_7(self, BMI):
        self.result += "\r\n Vysoké hodnoty BMI " + str(BMI)
        self.declare(PredictionBMI('bmi_checked'))

    @Rule(NOT(PredictionBMI('bmi_checked')),
          BodyMassIndexFromPatient(MATCH.BMI),
          TEST(lambda BMI: BMI is not None),
          BodyMassIndex(overweight_min=MATCH.overweight_min,
overweight_max=MATCH.overweight_max),
          AND(TEST(lambda overweight_min, BMI: BMI >= overweight_min),
              TEST(lambda overweight_max, BMI: BMI <= overweight_max)))
    def detect_disease_8(self, BMI):
        self.result += "\r\n Vyšší hodnoty BMI " + str(BMI)
        self.declare(PredictionBMI('bmi_checked'))

    @Rule(NOT(PredictionBMI('bmi_checked')),
          BodyMassIndexFromPatient(MATCH.BMI),
          TEST(lambda BMI: BMI is not None),
          BodyMassIndex(normal_weight_min=MATCH.normal_weight_min,
normal_weight_max=MATCH.normal_weight_max),
          AND(TEST(lambda normal_weight_min, BMI: BMI >=
normal_weight_min),
              TEST(lambda normal_weight_max, BMI: BMI <=
normal_weight_max)))
    def detect_disease_9(self, BMI):
        self.result += "\r\n Normální hodnoty BMI " + str(BMI)
        self.declare(PredictionBMI('bmi_checked'))

    @Rule(NOT(PredictionBMI('bmi_checked')),
          BodyMassIndexFromPatient(MATCH.BMI),
          TEST(lambda BMI: BMI is not None),
          BodyMassIndex(underweight=MATCH.underweight),
          TEST(lambda underweight, BMI: BMI < underweight))
    def detect_disease_10(self, BMI):
        self.result += "\r\n Nízké hodnoty BMI " + str(BMI)
        self.declare(PredictionBMI('bmi_checked'))

    @Rule(AND(Prediction('matches_all_symptoms'),
Prediction('DLCO_decreased'), Prediction('FEV1_decreased'),
              Prediction('SpO2_decrease'), PredictionBMI('bmi_checked'),
Prediction('smoker')))
```

```
    def detect_disease_11(self):
        self.result += "\r\n <strong> Pacient splňuje veškeré požadavky
pro nemoc: Emphysema </strong>"
```

Samostatný kód aplikace je možné zobrazit na stránkách https://bitbucket.org/HoloMaster/emphysema/src/master/ a zároveň je přiložen ve formátu zip k této práci.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Jan Horáček**
Studium:	I1900280
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Expertní sytém v Pythonu**
Název diplomové práce AJ:	Expert System in Python

Cíl, metody, literatura, předpoklady:

Cílem práce je navrhnout expertní systém, propojující bázi pravidel a neuronovou síť a realizovat ukázkovou aplikaci v CLIPSu a Pythonu.

1) Úvod
2) Teoretická část
- Expertní systémy
- Neuronové sítě
- CLIPS
- Python
- Přehled aplikací
3) Praktická část
- Diagnostika emfyzému
- Návrh báze pravidel
- Návrh rozhraní
- Popis implementace
4) Výsledky
5) Závěr

- Braido, F., Santus, P., Corsico, A. G., Di Marco, F., Melioli, G., Scichilone, N., {\& Solidoro, P. (2018). Chronic obstructive lung disease "expert system": validation of a predictive tool for assisting diagnosis. *International journal of chronic obstructive pulmonary disease, 13, 1747–1753.*
- *Xiaomin, Pei. (2015). Emphysema Classification Using Convolutional Neural Networks. 9244. 455-461.*
- *https://www.python.org/*
- *http://clipsrules.sourceforge.net/*
- *https://paperswithcode.com/*

Garantující pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	doc. RNDr. Kamila Štekerová, Ph.D.
Datum zadání závěrečné práce:	21.1.2020