# Zadanie číslo 2

Eduardo Milan Martinez

Obsah

[Zadanie číslo 2 1](#_Toc24677813)

[Zoznam použitých knižníc 1](#_Toc24677814)

[Úloha 1) - Načítanie dát a ich spracovanie 1](#_Toc24677815)

[Úloha 2) – Viacvrstvový perceptrón pre klasifikáciu 2](#_Toc24677816)

[Porovnanie rôznych architektúr: 2](#_Toc24677817)

[Úloha 3) – Regresný klasifikátor 5](#_Toc24677818)

[Porovnanie rôznych architektúr 5](#_Toc24677819)

## Zoznam použitých knižníc

* pandas
* numpy
* sklearn
* matplotlib
* scipy
* keras

## Úloha 1) - Načítanie dát a ich spracovanie

Na načítanie dát z csv súboru som použil dataframe z pythonovskej knižnice pandas. Keďže náš dataset je rozhádzaný a nie všetky údaje sú kompletné a použiteľné, musel som v prvom kroku vybrať užitočné stĺpce, ktoré sa dajú pri klasifikácii využiť. Medzi tieto stĺpce som zaradil dáta, ktoré by mohli byť užitočné pre klasifikáciu cenovej kategórie: napr. súradnice, geografická oblasť, superhost, počet recenzií, vybavenie bytu atď. Dáta obsahovali prázdne bunky. Prázdne miesta som vyplnil nulou. Kategorické dáta som sa rozhodol zakódovať pomocou LabelEncoderu z knižnice scikit learn. Ďalším krokom bolo vytvoriť cenové kategórie podľa ktorých prebieha klasifikácia. Na tvorbu kategórií som nekombinoval viaceré stĺpce, ale využil som iba samotný stĺpec ceny - ‘price‘. Vytvoril som tri kategórie, tak aby bolo množstvo dát rozložené rovnomerne medzi všetkými a uložil ich do výstupného stĺpca ‘price\_cat‘. Keďže keras NN potrebuje výstupné dáta ‘onehot encoded‘, musel som ešte tento stĺpec rozdeliť na tri pomocou funkcie ‘get dummies‘. Tieto stĺpce obsahujú iba jednotky a nuly.

Následným krokom bolo rozdelenie dát na množinu trénovaciu a testovaciu. Po rozdelení je potrebné dáta normalizovať. Na normalizáciu som použil StandartScaler z knižnice scikit learn, ktorý nám dáta transformuje do blízkosti rozmedzia od -2 do 4. Pri RBF sieti som využil MinMaxScaler, ktorý nám dáta transformuje medzi 0-1.

* **Vstupy**: dáta v CSV súbore
* **Výstupy**: upravené, normalizované, nenulové dáta, ktoré sú použiteľne pre strojové učenie
* **Metódy**: normalize\_data() – metóda, v ktorej sa upravujú dáta

## Úloha 2) – Viacvrstvový perceptrón pre klasifikáciu

Pre implementáciu viac vrstvového perceptróna som využil knižnicu Keras. Pôvodne som začínal s knižnicou scikit learn, ale nenašiel som dostatok informácií ako vybrať presnosť klasifikácie a stratu počas jednotlivých epoch tréningu.

Rýchlosť učenia som príliš nemanipuloval keďže pre väčšinu pokusov som využil solver ‘adam‘, s ktorým som dosahoval najvyššiu úspešnosť.

Aktivačná funkcia s ktorou som dosahoval najvyššiu úspešnosť bola funkcia relu.

Aby som zabránil pretrénovaniu, som pri trénovaní využil callback Early Stopping. V knižnici Keras callback predstavuje množinu funkcii, ktoré sú aplikované počas priebehu trénovania. Callback Early Stopping mi zabezpečí, že sa trénovanie zastaví, ak sa istý parameter prestane zlepšovať. Pre zastavenie trénovanie som sledoval parameter ‘validation\_loss‘.

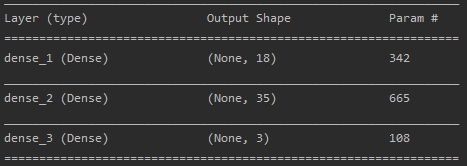
Pre validáciu počas trénovania som využil 15% dát z trénovacej množiny.

* **Vstupy**: trénovacie, testovacie dáta
* **Výstupy**: klasifikované vzorky, presnosť a strata počas tréningu
* **Metódy**: run\_mlp()

### Porovnanie rôznych architektúr:

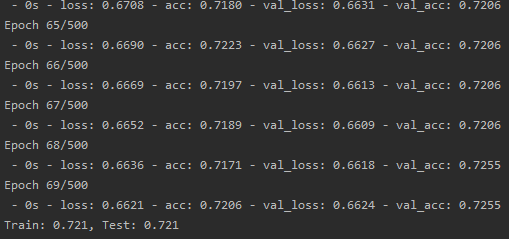
Ako prvý príklad uvádzam MLP s nasledovnou architektúrou:

* Počet vrstiev: 3
* Počet neurónov: Vstupná vrstva – počet neurónov = počet ukazovateľov (18), Skrytá vrstva = 35, Výstupná vrstva = počet klasifikačných tried (3)
* Aktivačná funkcia: relu, relu, softmax
* Solver: Adam
* Learning Rate: 0.0001
* Počet epoch – nastavený na 500, avšak kvôli predčasnému ukončeniu túto hodnotu nedosiahneme
* Zastavovacia podmienka: povolená

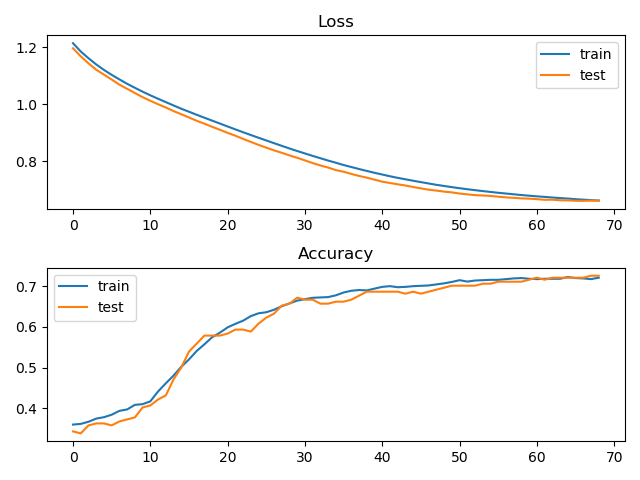


Obrázok 1 - Architektúra NS

Úspešnosť klasifikácie daného modelu na testovacích dátach bola 72,1%, čo môžeme považovať za pozitívny výsledok. Priebeh presnosti klasifikácie a straty na tréningových aj testovacích dátach je možné vidieť na grafe. Úspešnosť by sa dala zväčšiť lepším výberom validačnej množiny – v tomto prípade som zvolil ako validačnú množinu posledných 15% tréningových dát.



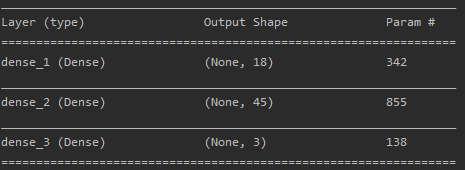
Obrázok 2 - Záverečné epochy a výsledok



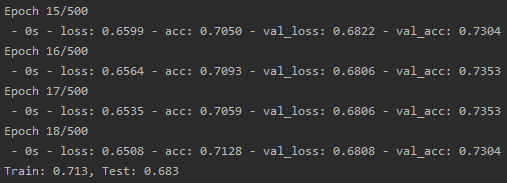
Obrázok 3 - Graf presnosti a straty klasifikácie

Ako druhý príklad uvádzam MLP s nasledovnou architektúrou:

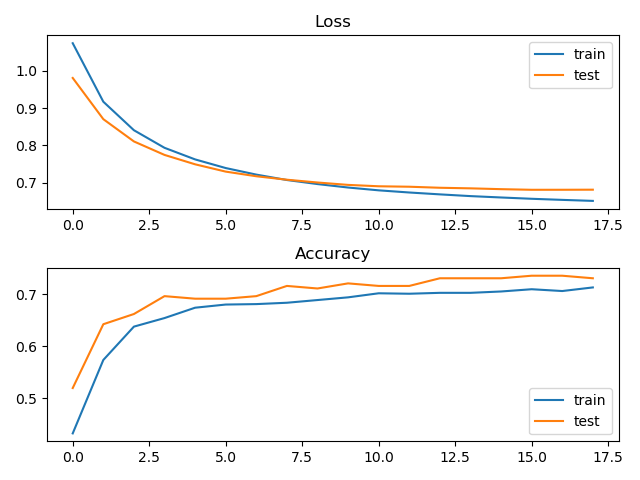
* Počet vrstiev: 3
* Počet neurónov: Vstupná vrstva – počet neurónov = počet ukazovateľov (18), Skrytá vrstva = 40, Výstupná vrstva = počet klasifikačných tried (3)
* Aktivačná funkcia: tanh, tanh, softmax
* Solver: SGD
* Learning Rate: 0.01
* Počet epoch – nastavený na 500, avšak kvôli predčasnému ukončeniu túto hodnotu nedosiahneme
* Zastavovacia podmienka: povolená



Obrázok 4 - Architektúra NS



Obrázok 5 - Záverečné epochy a výsledok



Obrázok 6 - Graf presnosti a straty klasifikácie

Po zmene solvera, aktivačných funkcií, počtu neurónov a rýchlosti učenia dostávame podobný výsledok. Úspešnosť klasifikácie daného modelu na testovacích dátach bola 68,3%. Priebeh presnosti klasifikácie a straty na tréningových aj testovacích dátach je možné vidieť na grafe.

## Úloha 3) – Regresný klasifikátor

Ako regresný klasifikátor som využil RBF sieť, vytvorenú v knižnici Keras. RBF sieť dosahovala lepší výsledk pri väčšom počte dát.

Pre testovanie som využil 15% z celkového počtu dát. Rovnaké množstvo dát som využil aj na validáciu počas trénovania siete.

Aby som zabránil pretrénovaniu, som pri trénovaní opäť využil callback Early Stopping. Callback Early Stopping mi zabezpečí, že sa trénovanie zastaví, ak sa istý parameter prestane zlepšovať. Pre zastavenie trénovanie som sledoval parameter ‘validation\_loss‘.

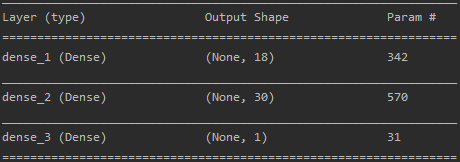
Na evaluáciu výsledkov som použil tri metriky. Mean Squared Error(MSE), Mean Absolute Error(MAE) a R2-Score.

* **Vstupy**: normalizované dáta
* **Výstupy**: predikcia ceny, evaluácia pomocou metrík
* **Metódy**: run\_rbf()

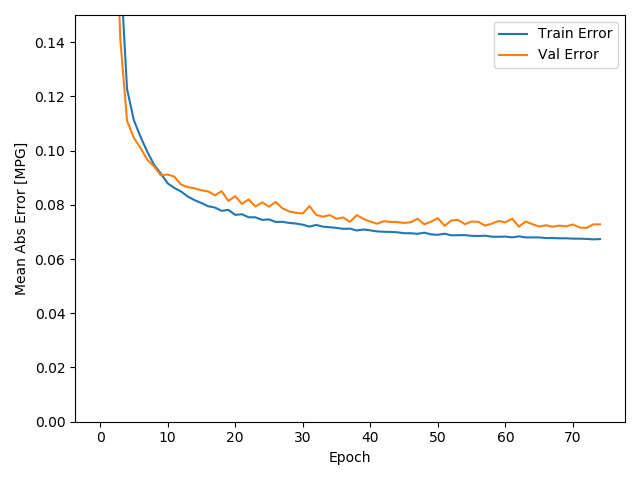
### Porovnanie rôznych architektúr

Ako prvý príklad uvádzam RBF s nasledovnou architektúrou:

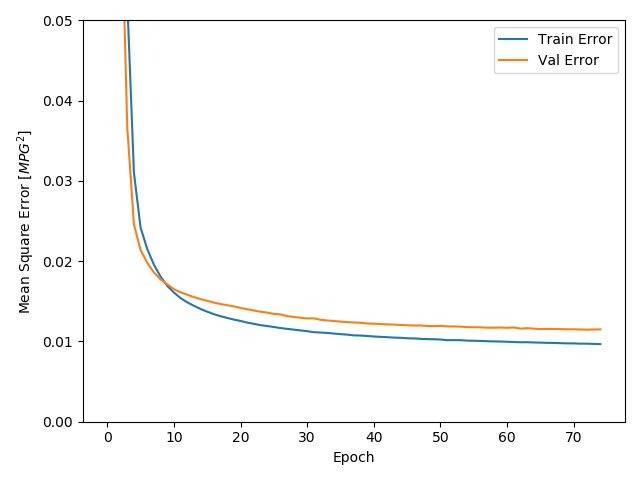
* Počet vrstiev: 3
* Počet neurónov: Vstupná vrstva – počet neurónov = počet ukazovateľov (18), Skrytá vrstva = 30, Výstupná vrstva = predikovaná cena (1)
* Aktivačná funkcia: relu, relu, linear
* Solver: RMSprop
* Learning Rate: 0.0001
* Počet epoch – nastavený na 1000, avšak kvôli predčasnému ukončeniu túto hodnotu nedosiahneme
* Zastavovacia podmienka: povolená



Obrázok 7 - Model NS



Obrázok 8 - MAE



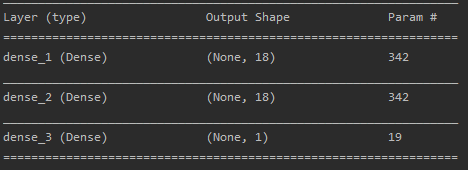
Obrázok 9 – MSE

Na validáciu som použil tri metriky. Výsledný Mean Squared Error na testovacích dátach bol v tomto prípade 0.108. Mean Absolute Error bol v tomto prípade 0.117. Poslednou validačnou technikou je R2-Score, ktorej výsledkom bola hodnota 0.526. Priebeh daných metrík počas trénovania je zobrazených na grafoch.

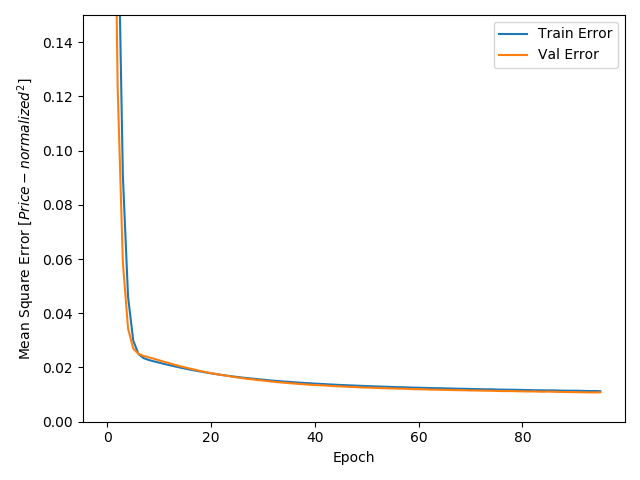


Ako druhý príklad uvádzam RBF s nasledovnou architektúrou:

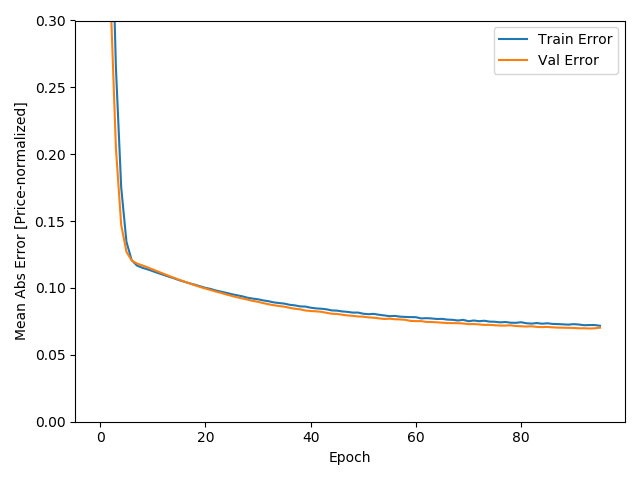
* Počet vrstiev: 3
* Počet neurónov: Vstupná vrstva – počet neurónov = počet ukazovateľov (18), Skrytá vrstva = 18, Výstupná vrstva = predikovaná cena (1)
* Aktivačná funkcia: tanh, tanh, linear
* Solver: Adam
* Learning Rate: 0.0001
* Počet epoch – nastavený na 1000, avšak kvôli predčasnému ukončeniu túto hodnotu nedosiahneme
* Zastavovacia podmienka: povolená



Obrázok 10 - Model NS

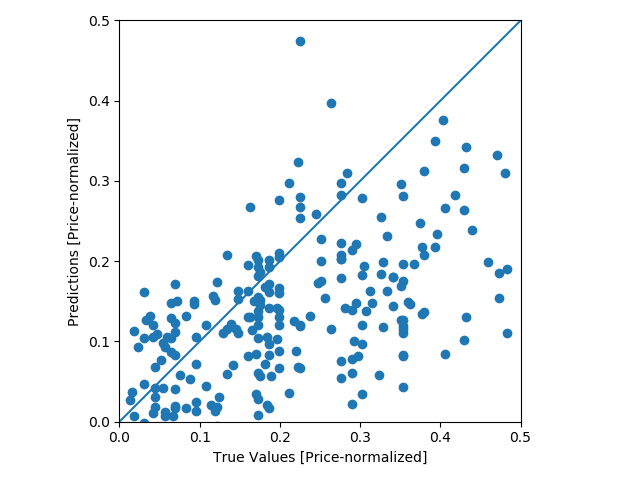


Obrázok 11 - MSE



Obrázok 12 – MAE

Pri danom modely môžeme pozorovať, že prišlo k voľbe príliš veľkej rýchlosti učenia. Zakryvenie krivky je príliš strmé. MSE na tréningových dátach je v tomto prípade 0.107 a testovacích 0.178. R2-Score dosahuje nižšiu hodnotu – 0.095.



Obrázok 13 - R2