I-SUNS: Zadanie č.1

Neurónové siete – JupyterLab Python

Malý predslov

Dokumentáciu som písal postupne ako som prechádzal cez zadanie, teda postupne sa tam nachádzajú aj nedostatky, ktoré boli neskôr pridané a teda sú údaje presnejšie a viac správne ako by mali byť. Využité boli knižnice alebo moduly ako Pandas, Matplotlib, Random, Numpy, Sys, H5py, Os, Tensorflow, Keras, Sklearn. Ďakujem za trpezlivosť pri čítaní. Zdroj z JupyterLab je Untitled.ipynb ten druhý súbor .py je len prekopírovaný odtiaľ, nevedel som či stačí len ten jeden.

Načítanie dát

Dáta **wine_test** pred normalizáciou a po

```
normalized_wine_test=(wine_test-wine_test.min())/(wine_test.max()-wine_test.min())
```

	Priemerná hodnota		**Štandardná odchýlka**	
	Pred	Po	Pred	Po
type	0.246923	0.246923	0.431388	0.431388
fixed acidity	7.258122	0.268256	1.337632	0.117336
volatile acidity	0.339483	0.207587	0.165460	0.132368
citric acid	0.328791	0.405915	0.143500	0.177160
residual sugar	5.375769	0.187653	4.679869	0.183885
chlorides	0.056916	0.074985	0.039065	0.065217
free sulfur dioxide	29.850000	0.263235	17.076848	0.167420
total sulfur dioxide	116.038462	0.325558	56.617037	0.167506
density	0.994730	0.464693	0.002902	0.184472
pH	3.213408	0.346103	0.161595	0.153900
sulphates	0.536905	0.165841	0.156280	0.090335
alcohol	10.457954	0.367492	1.191059	0.212689
quality	0.628462	0.628462	0.483402	0.483402

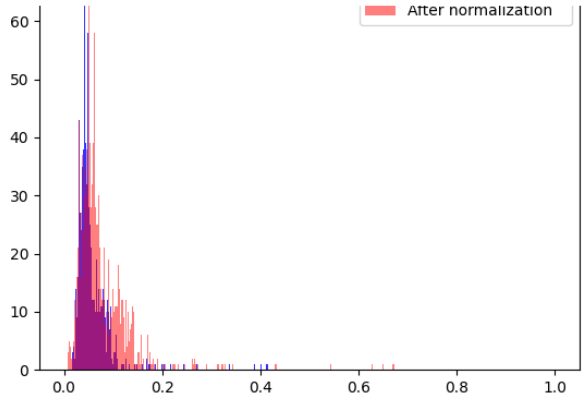
Dáta **wine_train** pred normalizáciou a po

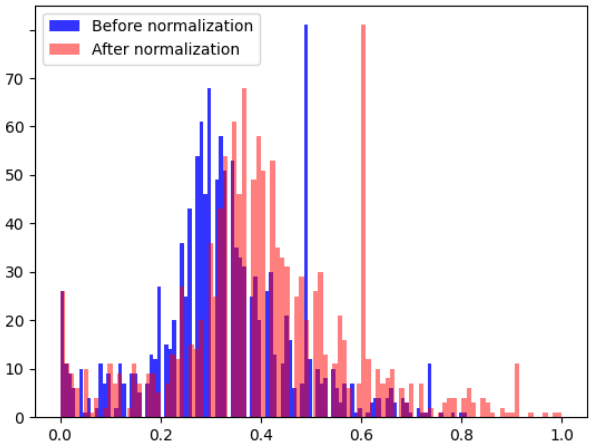
```
normalized_wine_train=(wine_train-wine_train.min())/(wine_train.max()-wine_train.min())
```

	Priemerná hodnota		**Štandardná odchýlka**	
	Pred	Po	Pred	Po
type	0.245911	0.245911	0.430668	0.430668
fixed acidity	7.206178	0.281502	1.284984	0.106197
volatile acidity	0.339743	0.173162	0.164335	0.109556
citric acid	0.316204	0.190484	0.145566	0.087690
residual sugar	5.461482	0.074563	4.776879	0.073265
chlorides	0.055823	0.077908	0.033947	0.056484
free sulfur dioxide	30.694247	0.103105	17.911303	0.062192
total sulfur dioxide	115.671060	0.252698	56.503236	0.130192

Dávid Gavenda 98533

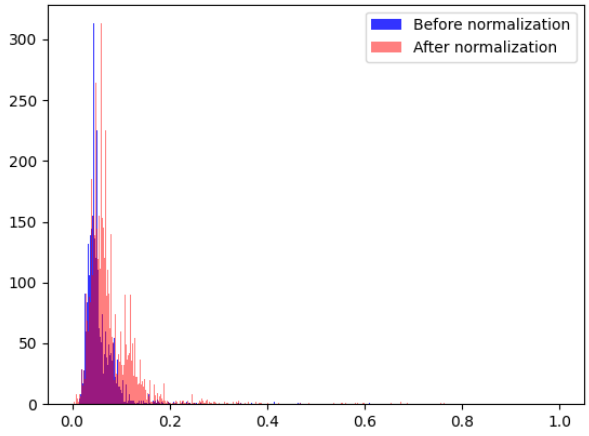
Histogramy – wine_test



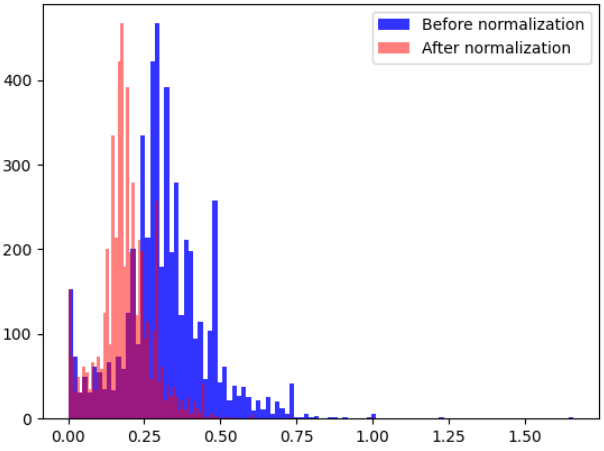


	Chlorides		Citric acid	
density	0.994688	0.146103	0.003023	0.058274
pH	3.219645	0.387322	0.160388	0.124332
sulphates	0.529792	0.174040	0.146810	0.082478
alcohol	10.500267	0.362358	1.193089	0.172911
quality	0.634212	0.634212	0.481697	0.481697

Histogramy – wine_train



Chlorides



Citric acid

Dávid Gavenda 98533

Náhodný klasifikátor zdroj

Nebol som si istý ako pristupovať k tomuto problému, napadli ma dve možnosti. V prvej som jednoducho náhodne generoval číslo 0 alebo 1, oboje s 50% pravdepodobnosťou a porovnával s kvalitou, ktorá bola zadaná pre testovacie dáta. Po zbehnutí testov 1000x bola hodnota 50.00% teda teória sedí.

Alebo z trénovacej množiny som vyrátal počet 1 a 0. Na základe toho nastavil pravdepodobnosť ich výskytu v dátach testovacích. Pravdepodobnosť pre 1 bola 0.6285 a pre 0 to bolo 0.3715. Keď som zbehol celý cyklus 1000x a overoval náhodné čísla s novou pravdepodobnosťou úspešnosť bola 53.20% čo je lepšie avšak zanedbateľne.

$$\begin{aligned} acc &= P(class=0) * P(prediction=0) + P(class=1) * P(prediction=1) \\ &= (0.6285 * 0.6285) + (0.3715 * 0.3715) \\ &= 0.5330 \end{aligned}$$

Úspešnosť by na sa základe vzorca mala blížiť k hodnote 53.30% čo sa aj blíži a pri väčšom počte opakovaní by ju pravdepodobne dosiahla.

Logistická regresia zdroj

Náš dataset má chýbajúce hodnoty, tieto sme nahradili priemerom (z trénovacej). Výsledok, ktorý chceme je binárny, teda 0 alebo 1. Máme viac ako 50 vzorkov, v prípade testovacích dát je ich 1300 čo postačuje. Skóre na základe logickej regresie je 0.75 čo znamená že dataset je dobrý. Čím je číslo bližšie k 1 tým lepšie. Rozdelili sme dataset na dve časti, kde jedna obsahovala kvalitu a druhá všetko okrem kvality, následne na všetko okrem kvality sme použili *scale*.

0.75

	precision	recall	f1-score	support
0	0.70	0.58	0.63	483
1	0.77	0.85	0.81	817
accuracy			0.75	1300
macro avg	0.74	0.72	0.72	1300
weighted avg	0.75	0.75	0.74	1300

Classification report

Neurónová sieť zdroj, graf

Prvá [keras]

!!!Dôležitá poznámka : Chýba tu validačné rozdelenie...!!!

```
model.add(Dense(16, input_dim=12,activation='relu'))
model.add(Dense(12,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=[,accuracy'])
model.fit(X, Y, epochs=150, batch_size=10)
```

Skrytá vrstva má 16 *nodes*. Náš model má 12 premenných. Druhá skrytá vrstva má 12 *nodes* a využíva „relu" aktivačnú funkciu. Využívame entropiu „loss", ktorá sa využíva na binárnu klasifikáciu problémov. *Optimizer* používa algoritmus „adam". Na trénovanie využijeme 150 epóch a *batch size* 12. Jedná sa o všeobecný vzorec, ktorý bol použitý na stránke v zdroji. Výsledkom bola presnosť 77.97%.

SGD	76,01
RMSprop	78,31
Adam	77,97
Adadelta	63,65
Adagrad	70,35
Adamax	75,91
Nadam	78,51
Ftrl	63,42

Po vyskúšaní iných *Optimizers* vyšiel najlepší "Nadam", všetky ostatné podmienky boli rovnaké.

Najlepšie výsledky (pokiaľ nie je spomenutá zmena, rovnako ako hore):

Nadam, 500 epóch , batch_size 50 79.10%

Nadam, 500 epóch , batch_size 75 79.33%

Nadam, 20 nodes, 1000 epóch , batch_size 75 79.49%

Nadam, 50 nodes, 1000 epóch , batch_size 75 82.30%

Nadam, 100 nodes, 1000 epóch , batch_size 75 85.70%

Nadam, 250 nodes, 1000 epóch , batch_size 75 91.78%

Nadam, 1500 nodes, 1000 epóch , batch_size 75 98.98%

Nadam, 5000 nodes, 1000 epóch , batch_size 75 99.73%

Nadam, 500 nodes, 10000 epóch , batch_size 75 99.96%

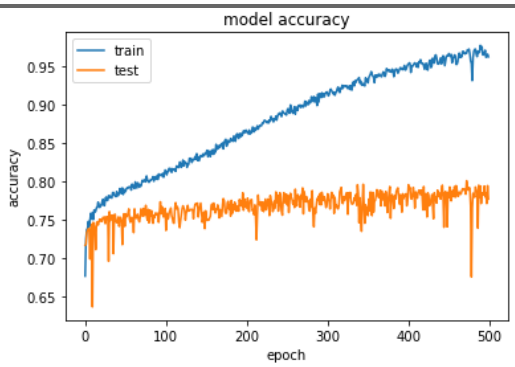
(neviem akým zázrakom ale už pri 2000. to išlo k 100%. Bežalo to asi 5 minút maximálne)

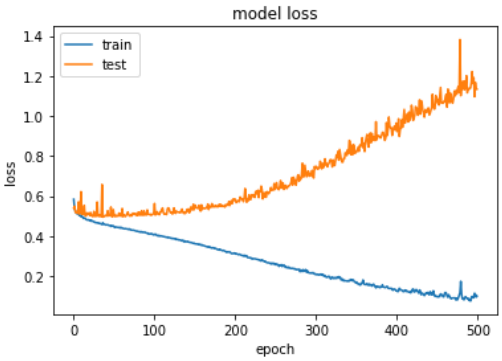
Dávid Gavenda 98533

Neskoršia poznámka “Už viem akým zázrakom :D”

Prvá s validačnou množinou

```
model.add(Dense(5000, input_dim=12,activation='relu'))
model.add(Dense(12,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='Nadam', metrics=['accuracy'])
history = model.fit(X, Y, validation_split=0.33, epochs=500, batch_size=75)
```





Presnosť: 91,21%

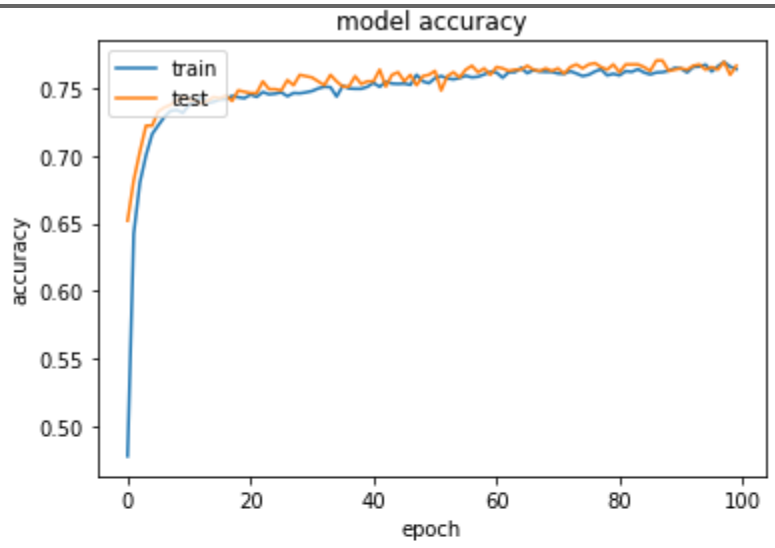
Pri použití menšej validačnej vrsty sme dosiahli presnejšie výsledky, napr. 0,2 mala úspešnosť 94%. Ďalším cieľom je nájsť rozumnú sieť a nie nejakú, ktorá má 5000 *nodes*...

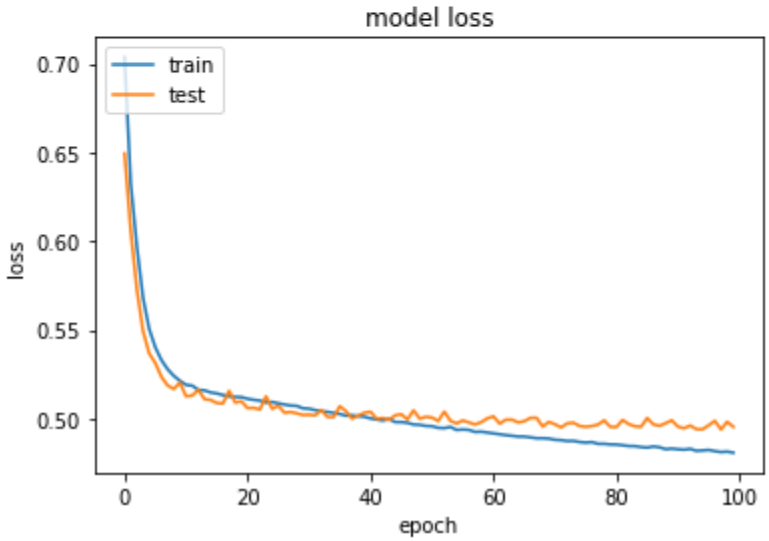
Dávid Gavenda 98533

Hľadanie efektívnej

Cieľom je dosiahnuť aspoň 75% úspešnosť.

```
model.add(Dense(25, input_dim=12,activation='relu'))
model.add(Dense(12,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='Nadam', metrics=['accuracy'])
history = model.fit(X, Y, validation_split=0.2, epochs=100, batch_size=75)
```





Pokračoval som v používaní „nadam“ keďže sa mi overil, použil som aj iné pre overenie možností. Presnosť nám postupne stúpa a strata klesá, úspešnosť sa zastavila na 76.97%

Dávid Gavenda 98533

Pocit uvedomenia zdroj matica

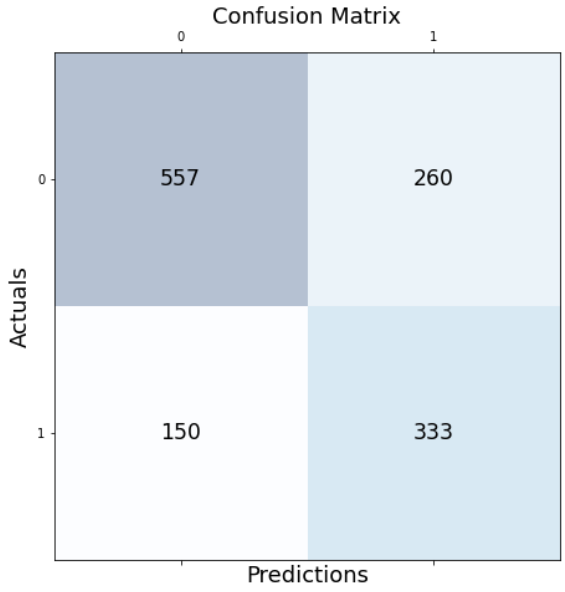
Celý čas som nepoužíval obe množiny ale len tu väčšiu z nich, to som teraz zmenil a využívam aj trénovacie aj testovacie dáta.

```
trainX = normalized_wine_train.loc[:, normalized_wine_train.columns != 'quality'].values
trainY = np.int64(normalized_wine_train['quality'].values)
testX = normalized_wine_test.loc[:, normalized_wine_test.columns != 'quality'].values
testY = np.int64(normalized_wine_test['quality'].values)
```

Pridal som *early stopping*, s celkom vysokou trpezlivosťou (200) a taktiež *model checkpoint*

```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=200)
mc = ModelCheckpoint('best_model.hdf5', monitor='val_accuracy', mode='max', verbose=1,
save_best_only=True)
```

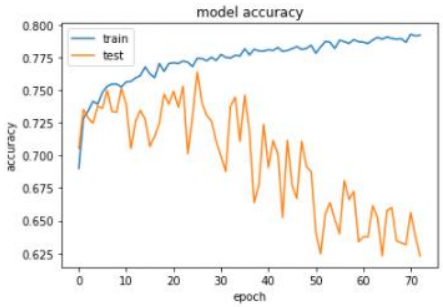
Najlepší model som si ukladal do 'best_model.hdf5', pomocou ktorého som získal taktiež hodnoty do *confusion matrix* a následne vypísať graf pre *confusion matrix. Prvotný vyzeral takto (nie je moc dobrý, presnosť bola okolo 68%.*

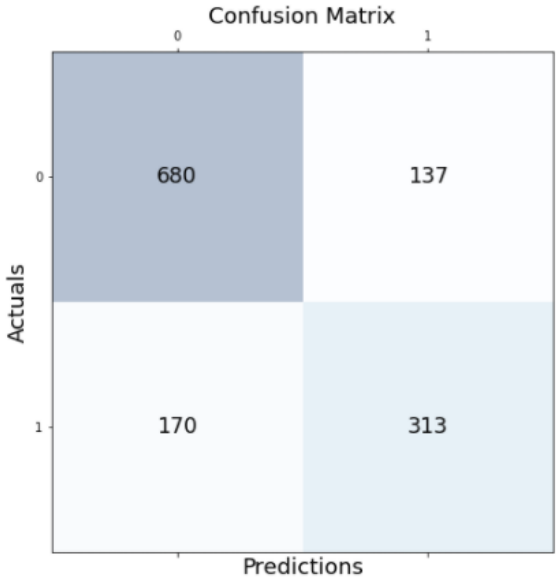


Neurónová sieť [zdroj na porovnávanie]

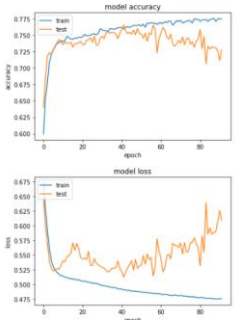
Po splnení všetkých výpisov nastal správny čas vycvičiť túto neurónovú sieť na minimálne 75% *úspešnosť. Prvý krát to bolo dosiahnuté s danou konfiguráciou.*

```
model.add(Dense(5000, input_dim=12,activation='relu'))
model.add(Dense(12,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='nadam', metrics=['accuracy'])
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=50)
mc = ModelCheckpoint(fileName, monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)
history = model.fit(trainX, trainY, validation_data=(testX, testY),  epochs=500, batch_size=75,
callbacks=[es, mc])
```

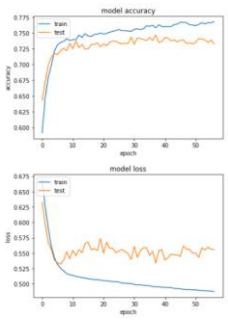




Presnosť bola 76.385%. Jednalo sa o postupne pomalé vylepšovanie, ktoré malo rovnocenné rozdelenie zlého odhadu. S výsledkom som spokojný. Pri znížení počtu *nodes* na 50 stúpla presnosť na 76.538%. Pri počte 25 klesla pod 75%. Môžeme si všimnúť že „test“ časť na grafoch je dosť skákavá, toto môžeme ovplyvniť znížením *learning rate.*



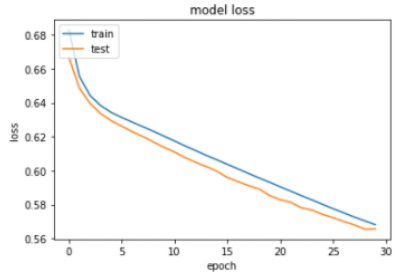
50 nodes



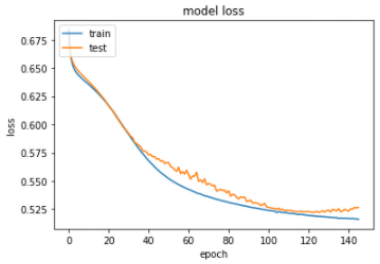
25 nodes

Dávid Gavenda 98533

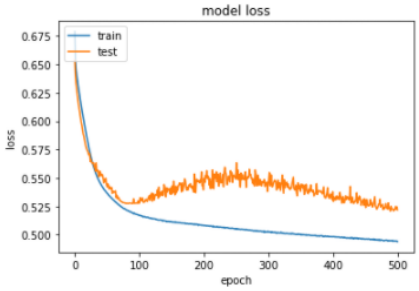
Vo väčšine prípadov sa zatiaľ jednalo o pretrénovanie, keďže *early stopping* mal nastavenú relatívne vysokú trpezlivosť. Jeho priveľkým znížením sa dostaneme na grafy, ktoré boli na druhú stranu pod trénované a dosahovali nízku úspešnosť. Správnym nastavením, dokážeme nájsť zlatú strednú cestu.



Pod trénovaná



Relatívne správne natrénovaná



Pretrénovaná

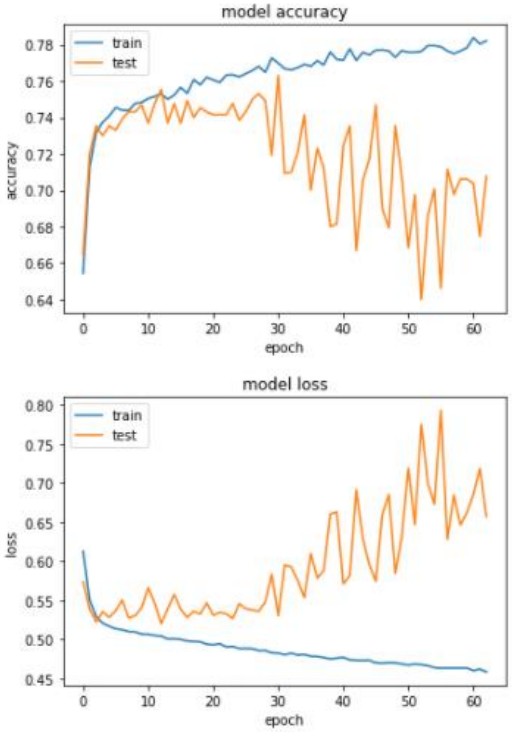
Najviac stabilné výsledky boli dosiahnuté s využitím nízkeho *learning rate*, bohužiaľ výsledky boli nižšie a preto som zvolil vyšší.

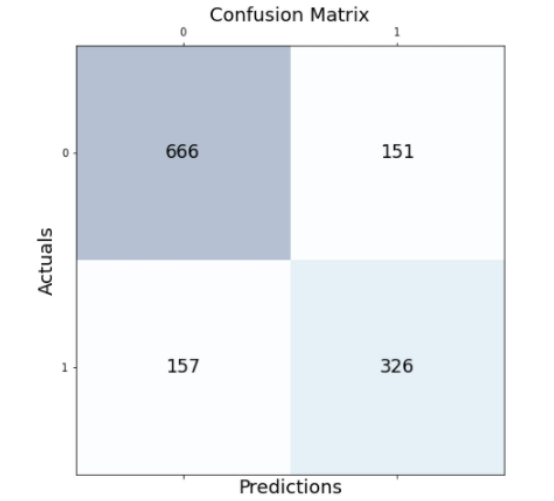
```
model.add(Dense(128, input_dim=12,activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(1,activation='sigmoid'))


keras.optimizers."Organizer"(
   learning_rate="Learning rate",
   name="Organizer"
)
model.compile(loss='binary_crossentropy', optimizer= "Organizer", metrics=['accuracy'])
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=50)
mc = ModelCheckpoint(fileName, monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)
history = model.fit(trainX, trainY, validation_data=(testX, testY),  epochs=500, batch_size=75, callbacks=[es, mc])
```

Ostatné nastavenia

Organizer/ Learning rate	25	10	1	0.01	0.0001	0.000001
SGD	0.73538	0.72154	0.72077	0.73692	0.73231	0.73615
RMSprop	0.75462	0.76308	0.76462	0.76846	0.76077	0.75077
Adam	0.75077	0.75846	0.75923	0.75077	0.75231	0.75538
Adadelta	0.65385	0.65154	-	-	-	0.66538
Adagrad	0.73154	-	-	-	-	0.72231
Adamax	0.73769	-	0.74846	-	-	0.74846
Nadam	0.74846	0.75077	0.75769	0.74923	0.75308	0.75077

- preskočené kvôli slabým výsledkom

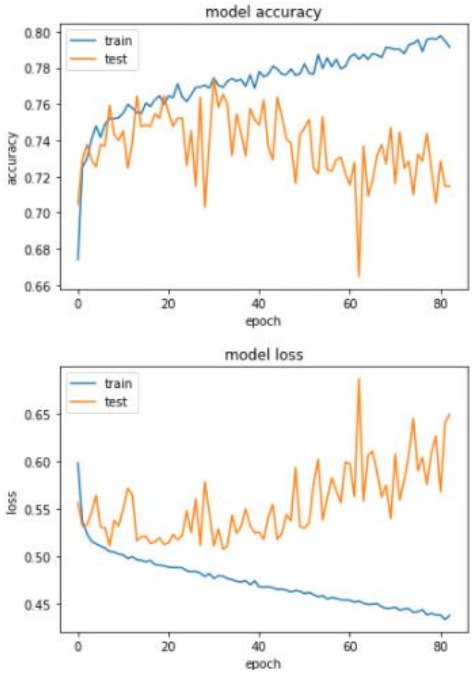


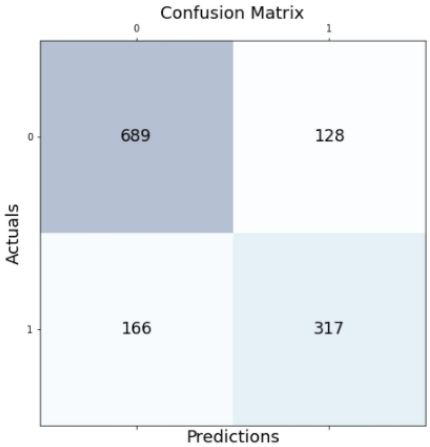


Classification report :

	precision	recall	f1-score	support
1	0.81	0.82	0.81	817
0	0.68	0.67	0.68	483
accuracy			0.76	1300
macro avg	0.75	0.75	0.75	1300
weighted avg	0.76	0.76	0.76	1300

Takto vyzerala zvolená s najlepšou úspešnosťou, pridaním ďalšej vrstvy úspešnosť jemne stúpla na 77.385% (*model.add(Dense(32,activation='relu'))*)





Classification report :

	precision	recall	f1-score	support
1	0.81	0.84	0.82	817
0	0.71	0.66	0.68	483
accuracy			0.77	1300
macro avg	0.76	0.75	0.75	1300
weighted avg	0.77	0.77	0.77	1300

BONUS – wine_test

	Median	
	Pred	Po
type	0.000000	0.000000
fixed acidity	0.245614	0.245614
volatile acidity	0.176000	0.176000
citric acid	0.395062	0.395062
residual sugar	0.094303	0.094303
chlorides	0.060100	0.060100
free sulfur dioxide	0.245098	0.245098
total sulfur dioxide	0.331361	0.331361
density	0.475524	0.475524
pH	0.333333	0.333333
sulphates	0.150289	0.150289
alcohol	0.339286	0.339286
quality	1.000000	1.000000

Sú rovnaké

Kvartily

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides
0.25	0.0	6.4	0.22	0.26	1.8	0.038
0.50	0.0	7.0	0.30	0.32	3.0	0.048
0.75	0.0	7.7	0.41	0.41	7.9	0.066

	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0.25	16.0	78.0	0.9924	3.10	0.44	9.5	0.0
0.50	28.0	118.0	0.9949	3.20	0.51	10.3	1.0
0.75	41.0	156.0	0.9970	3.32	0.60	11.3	1.0

Dávid Gavenda 98533