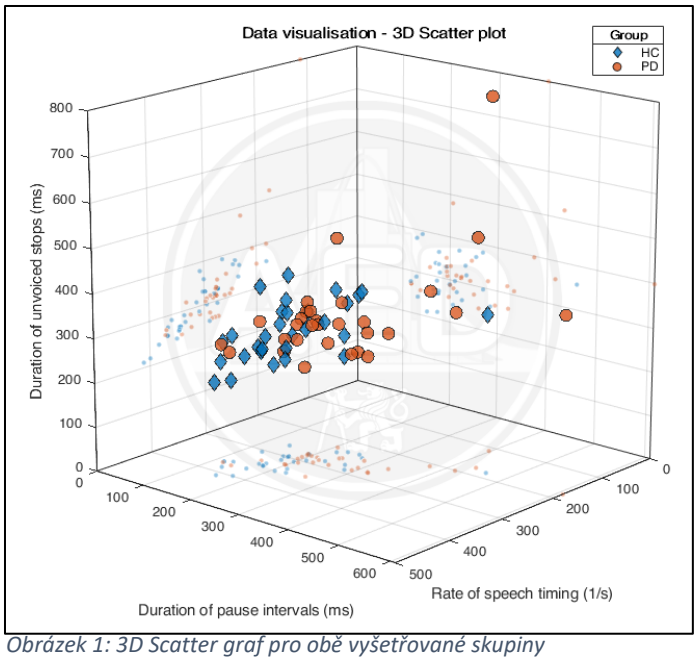
Zadání Cvičení #10

Popis dat: Pracovní data jsou uložena v souboru data.csv, který je k dispozici ke stažení na Moodle stránce tohoto předmětu, ve složce příslušného cvičení. Pro načtení dat do Matlabu využijte funkce `readtable`.

Data jsou ve formátu tabulky, která obsahuje data od pacientů s Parkinsonovou nemocí (label **PD**) a data od kontrolní skupiny zdravých lidí (label **HC**). Tabulka obsahuje ID kódy subjektů, identifikátory příslušnosti ke skupině (labely) a hodnoty těchto tří parametrů:

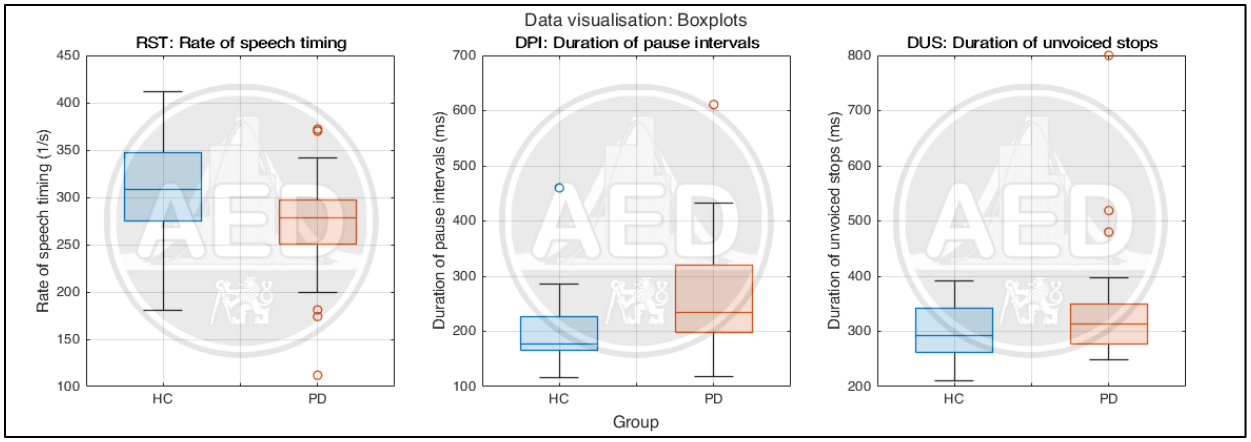
- **RST** – *Rate of Speech Timing*
  - Popisuje rychlost řeči pomocí identifikace znělých, neznělých a přerušených segmentů v záznamu. Jednotkou je s^{-1}.
- **DPI** – *Duration of pause intervals*
  - Popisuje kvalitu časování řeči (pauzy v řeči indikují problémy s iniciací), jednotkou jsou milisekundy, *ms*.
- **DUS** – *Duration of unvoiced stops*
  - Parametr popisující délku neznělých souhlásek, při kterých je náhle zastaven (a vypuštěn) proud vzduchu, jednotkou jsou opět milisekundy, *ms*.



Obrázek 1: 3D Scatter graf pro obě vyšetřované skupiny

Obě skupiny podstoupily vyšetření řeči pomocí volného monologu. Řečové nahrávky byly analyzovány a ze záznamu byly vypočteny hodnoty parametrů. Další informace viz Hlavnička et al. 2017 (Reference).

Pro zjednodušení práce **jsme za vás provedli vizualizaci dat** (viz grafy na této straně). Na *obrázku 1* můžete vidět vykreslená data pomocí 3D grafu typu scatter, včetně projekcí dat do rovin *x-y, y-z* a *x-z*. Tyto projekce pak přirozeně odpovídají 2D scatter grafům pro jednotlivé dvojice parametrů. *Obrázek 2* pak obsahuje klasické boxploty pro všechny tři parametry.



Obrázek 2: Boxploty pro analyzované parametry

Zadání úlohy	body

Z poskytnutých dat **natrénujte dva klasifikační modely** na standardizovaných datech (u obou modelů ponechte defaultní nastavení Matlabu pro funkci `fitcsvm`):
• SVM klasifikátor s **Lineárním Kernelem**
• SVM klasifikátor s *Radial Basis Function* **(RBF) Kernelem**
Oba modely validujte na stejných datech, ze kterých jste model natrénovali:
• Srovnejte skutečné (*true*) labely a predikované labely pomocí *Confusion Matrix*
  ○ Využijte funkce `confusionmat` pro výpočet a `confusionchart` pro zobrazení výsledků.
  ○ Nezapomeňte uvést správné označení skupin.
  ○ V tabulce si také můžete zobrazit přesnosti klasifikace pro jednotlivé skupiny pomocí argumentu `RowSummary`.
• Vykreslete si ROC (*Receiver operating characteristics*) křivku pro každý z modelů do jednoho obrázku.
  ○ Použijte funkci `perfcurve` (Jako pozitivní skupinu berte pacienty s PN).
  ○ Uveďte také hodnotu AUC pro oba modely.
Vykreslete obrázek s vypočtenými maticemi zmatení (Confusion Matrices, *zmaticemi*) a dobře ho popište. Také vykreslete obrázek s ROC křivkami pro oba modely. Slovy zapište hodnoty AUC a vašimi slovy zhodnoťte výsledky.
	1.5

Proveďte *5-Fold Cross-Validaci* SVM klasifikátoru s lineárním kernelem. Cross-validujte SVM s lineárním kernelem, model vždy učte na standardizovaných datech. Určete přesnost modelu na základě těchto veličin (hodnoty *True/False Positive/Negative* můžete určit manuálně nebo například pomocí funkce `confusionmat`): • Senzitivita • Specificita • Accuracy Specificity $SPC = \frac{TN}{TN + FP}$ Sensitivity $SEN = \frac{TP}{TP + FN}$ Accuracy $ACC = \frac{TP + TN}{TP + FP + FN + TN}$ **Způsob výpočtu 5-fold cross-validace lze provést** (libovolně vyberte): • Manuálním rozdělením na testovací a trénovací množiny pro 5-fold schéma, iterativním výpočtem SVM modelu a průměrováním výsledných hodnot senzitivity, specificity a accuracy z jednotlivých iterací. 😕 • Rozdělením dat pro 5-fold schéma pomocí funkce `cvpartition` a opět iterativním výpočtem modelu SVM a průměrováním výsledků. 😊 • Přímým použitím vestavěné cross-validace ve funkci pro výpočet SVM modelu, `fitcsvm`, a klasifikace pomocí funkce `kfoldPredict`. 😊 **Jedním z výše uvedených způsobů naimplementujte 5-fold cross-validaci. Uveďte výsledné vypočtené hodnoty senzitivity, specificity a accuracy.**	1

Najděte optimální hodnoty hyperparametrů SVM klasifikátoru s RBF kernelem - *Box constraint* a *Kernel scale* (v přednášce značené jako *C* a *σ*) pomocí techniky *Grid Search*.

- Mřížku pro techniku *Grid Search* nastavte v rozsahu hodnot **0.01 až 3.01** pro oba parametry, přičemž testovací krok zvolte jako **0.1**.
- Na každém bodu mřížky hyperparametrů vypočtěte hodnoty **specificity, senzitivity a accuracy** cross-validovaného modelu pomocí techniky *Leave-One-Out*.
  - Ať už jste v předchozím bodě realizovali cross-validaci jakkoli, zde využijte vstupního argumentu `Leaveout` funkce `fitcsvm`.
  - Výpočet modelů ve všech bodech mřížky může zabrat několik minut v závislosti na rychlosti vašeho počítače. Pokud máte nainstalovaný *Parallel Computing Toolbox*, můžete pro zrychlení výpočtu využít cyklu `parfor`, který automaticky paralelizuje výpočty na všechna procesorová jádra vašeho počítače.
- **Výsledné optimalizační povrchy vykreslete** pomocí funkce `plotGridSearch`, která je dostupná ke stažení na Moodlu.
  - Tato funkce vám také vrátí tabulku s optimálními hodnotami hyperparametrů z hlediska specificity, senzitivity a accuracy.
  - Popis toho, jak do funkce `plotGridSearch` vložit vstupní data, naleznete přímo v ní.

Implementujte optimalizaci parametrů pomocí techniky Grid Search. Vytvořte pomocí funkce `plotGridSearch` obrázek s vypočtenou mřížkou a zapište výsledné optimální body z mřížky (optimální hodnoty parametrů *Box Constraint* a *Kernel Scale* pro nejlepší hodnoty *senzitivity*, *specificity* a *accuracy*).

1.5

Nepovinný bonus:

Naprogramujte v Matlabu nejjednodušší neuronovou síť – Perceptron. 😲

Načtěte ze souboru bonus.mat data pomocí funkce `load`. Data v proměnných *A* a *B* jsou vektory o velikost 2x100, tedy dva soubory stovky 2D bodů. Tyto body mohou reprezentovat např. hodnoty dvou parametrů pro dvě rozdílné skupiny lidí.

Nejprve trochu teorie… 🙄

Uvažujte rozhodovací funkci $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$, která rozdělujte prostor na dvě poloroviny. Pokud $\boldsymbol{w} = [w_0, w_1, w_2]$ je normálový vektor k rozhodovací hranici a $\boldsymbol{x} = [1, x_1, x_2\,]$ je jeden datový bod, pak pokud:

- $g(\boldsymbol{x}) > 0$, $\boldsymbol{x}$ patří do třídy 1
- $g(\boldsymbol{x}) < 0$, $\boldsymbol{x}$ patří do třídy 2
- $g(\boldsymbol{x}) = 0$, $\boldsymbol{x}$ leží na rozhodovací hranici.

A navíc absolutní hodnota funkce $g(\boldsymbol{x})$ udává vzdálenost bodu od rozhodovací hranice.

Dále uvažujme funkci $f(\boldsymbol{w}, \boldsymbol{x}_i) = l_i \cdot g(\boldsymbol{w}, \boldsymbol{x}_i)$. Tato funkce „kontroluje“ správnou klasifikaci tím, že kombinuje údaj o skutečné příslušnosti (v labelu l_i) a výsledek klasifikace z funkce $g(\boldsymbol{w}, \boldsymbol{x}_i)$. Správně klasifikované body budou mít výsledek funkce $f(\boldsymbol{w}, \boldsymbol{x}_i)$ vždy kladný, nesprávně klasifikované pak vždy záporný.

0.5

Můžeme pak vytvořit chybovou funkci $\mathcal{J}_p$, která se nazývá ***Perceptron criterion function*** a která bude akumulovat vzdálenosti všech špatně klasifikovaných bodů od rozhodovací hranice (mínus je tam protože špatně klasifikované body budou mít zápornou hodnotu funkce a my pro ně chceme kladnou chybu):

$$\mathcal{J}_p(\boldsymbol{w}) = \sum_{\boldsymbol{x_i} \in \boldsymbol{X}} -f(\boldsymbol{w}, \boldsymbol{x}_i) = \sum_{\boldsymbol{x_i} \in \boldsymbol{X}} -l_i \cdot \boldsymbol{w}^T \boldsymbol{x}_i$$

Pokud tato funkce bude mít nulovou hodnotu, znamená to, že všechny vzorky byly správně klasifikovány – tím pádem je cílem jí minimalizovat.
Najdeme minimum – provedeme derivaci této funkce vzhledem k $\boldsymbol{w}$:

$$\nabla \mathcal{J}_p = \sum_{\boldsymbol{x_i} \in \boldsymbol{X}} -l_i \boldsymbol{x}_i$$

Tím pádem poté, co náhodně „nastřelíme“ rozhodovací hranici iniciací vektoru $\boldsymbol{w}$**,** se podíváme na momentální klasifikace všech vzorků. Vybere ty vzorky, které jsou špatně klasifikované a s jejich pomocí updatujeme naší rozhodovací hranici ve směru nižší hodnoty chybové funkce.

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \cdot \nabla \mathcal{J}_p = \boldsymbol{w}(t) + \eta \sum_{\boldsymbol{x_i} \in \boldsymbol{X}} l_i \boldsymbol{x}_i$$

Kde t značí číslo „epochy“, po kterou je rozhodovací hranice stejná a pracuje se na aktualizaci, a η značí „velikost kroku“, tedy jak rychle se bude hranice moci měnit skrze epochy. η je dobré nastavit např. na hodnotu 0.1, 0.01, 0.001 apod.

Prakticky…

- Obě skupiny **sjednoťte do vektoru** o velikost 2x200.
- **Rozšiřte sjednocený vektor dat vektorem samých jedniček**. Toto je důležité proto, aby algoritmus mohl vypočítat *bias* (váhu, která posouvá rozhodovací hranici).
- Připravte si vektor labelů $L = l_1, l_2, \dots, l_N$ o velikosti 1x200, který bude odpovídat vektoru dat, **s hodnotami $\mathbf{1}$ pro skupinu *A* a $\mathbf{-1}$ pro skupinu *B***.
- **Inicializujte si vektor vah** (*weights*) ***w*** o velikosti 3x1 náhodnými malými čísly (např. 0.1). Vektor vah má jeden prvek pro každou z dimenzí dat a jeden prvek pro výpočet *biasu* w_0. Také si nastavte hodnotu velikosti kroku η.
- **Implementujte algoritmus *batch perceptronu*** teoreticky popsaný výše a v pseudokódu shrnutý v rámečku.

Do Moodlu odevzdejte obrázek s vaší kódovou implementací Perceptronového klasifikátoru a grafické zobrazení dat a vypočtené rozhodovací hranice.

Algoritmus Hromadného (*Batch*) Perceptronu

1: Inicializujte hodnoty: **w**, η
2: **Iterujte epochy**:
3: $X = \{\}$
4: **Pro**: $i = 1{:}N$
5: **Pokud**: $f(\boldsymbol{w}, \boldsymbol{x_i}) < 0$, pak $x_i \rightarrow X$
6: $\boldsymbol{w} = \boldsymbol{w} + \eta \cdot \sum_{\boldsymbol{x_i} \in X} \boldsymbol{l_i x_i}$
7: **Dokud** $X = \{\}$

Reference

Hlavnička, J., Čmejla, R., Tykalová, T., Šonka, K., Růžička, E., and Rusz, J. (2017). *Automated analysis of connected speech reveals early biomarkers of Parkinson's disease in patients with rapid eye movement sleep behaviour disorder.* Scientific reports, 7, 12, https://doi.org/10.1038/s41598-017-00047-5.