

**Support Vector Machines s kernelmi**

SEMESTRÁLNA PRÁCA

Vypracoval(a): **Adam Hajro**

Študijná skupina: **5ZYI25**

Predmet: **Dáta, informácie, znalosti**

Cvičiaci: *Ing. Lukáš Falát, PhD.*

Obsah

[1. Support vector machines 3](#_Toc126858931)

[1.1. Čo sú vlastne zač? 3](#_Toc126858932)

[1.2. Lineárne dáta 3](#_Toc126858933)

[1.3. Nelineárne dáta 5](#_Toc126858934)

[1.4. Použitie kernelov 6](#_Toc126858935)

[1.5. Porovnanie SVM s kernelmi a bez kernelov 7](#_Toc126858936)

[2. Požitý dataset 7](#_Toc126858937)

[3. Postup implementácie vo vybranom jazyku 7](#_Toc126858938)

[4. Výsledky 8](#_Toc126858939)

# Support vector machines

## Čo sú vlastne zač?

“In machine learning, support vector machines (SVMs, also support vector networks[1]) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues (Boser et al., 1992, Guyon et al., 1993, Cortes and Vapnik, 1995,[1] Vapnik et al., 1997[citation needed]) SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

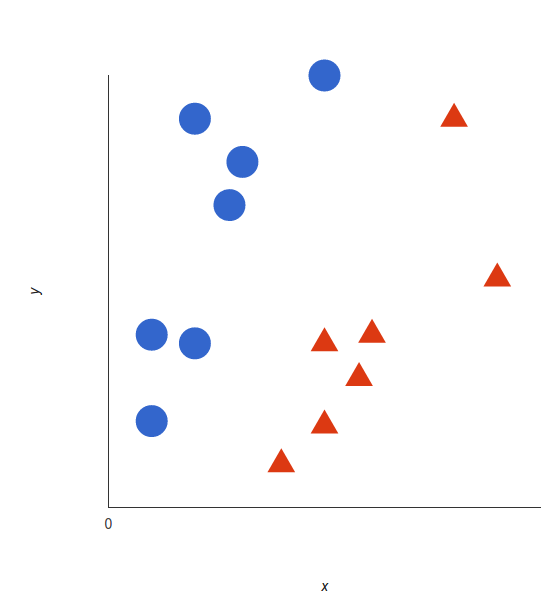
Jednoduché vysvetlenie:

„*In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Mostly are being used in classification problems*”

## Lineárne dáta

Základy fungovania SVM je možné pochopiť týmto jednoduchým príkladom.

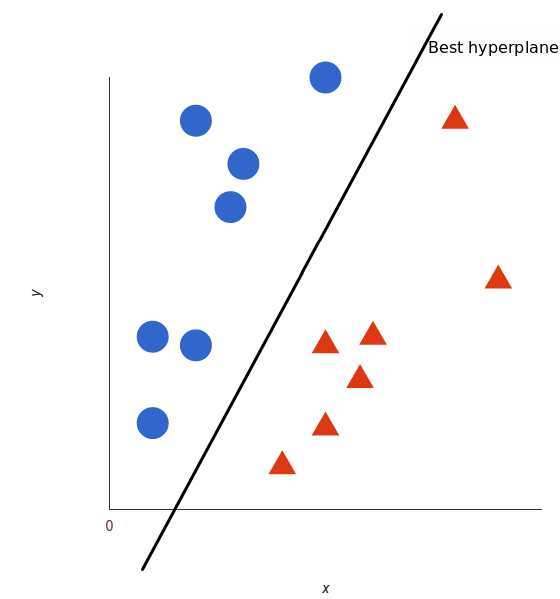
Predstavme si že máme

* dva tagy: **červený** a **modrý**
* a definované dáta sú na dvoch osách: **x** a **y**

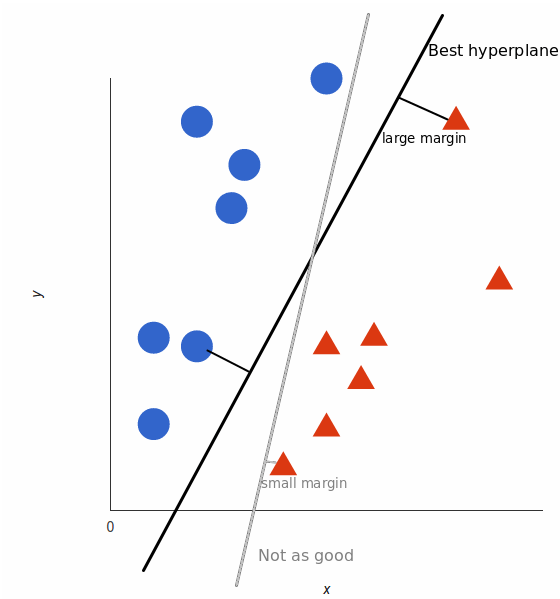
Chceme nájsť **jednoznačný** classifier, kt. na základe **x**, **y** súradníc určí či je výsledok **červený** alebo **modrý.**

SVM zoberie dátové body a vyznačí tzv. **hyperplane** (v 2D svete je to čiara), kt. najlepšie rozdeľuje tagy.

Táto čiara je rozhodujúca hranica:

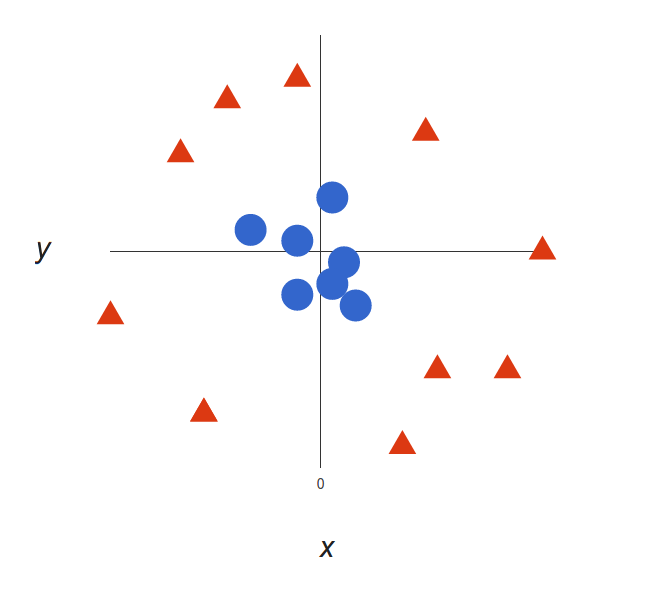
* všetko, čo padne na jej **JEDNU** stranu sa klasifikuje ako **MODRÉ**
* všetko, čo padne na jej **DRUHÚ** stranu sa klasifikuje ako **ČERVENÉ**

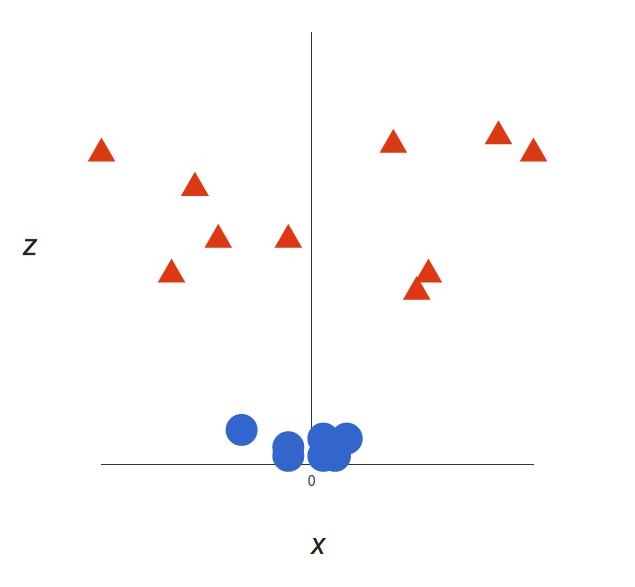
Potom tu nastáva otázka že kt. **hyperplane**/čiara je najlepšia?

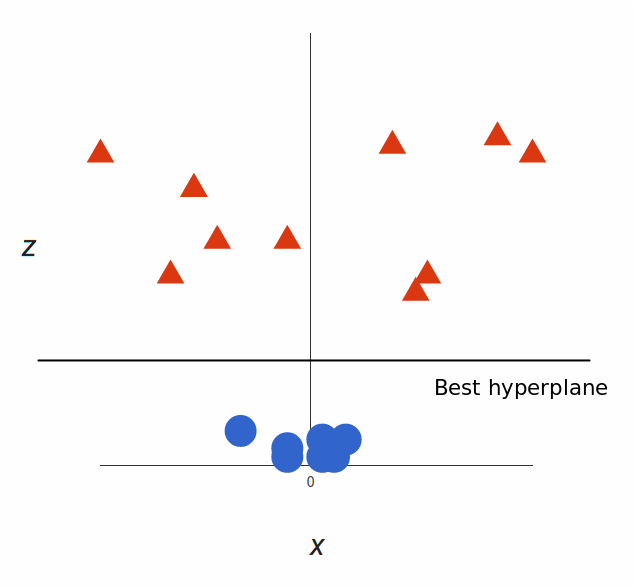
* pre SVM je to tá, kt. **maximalizuje** *margins* z obidvoch tagov
* resp. hyperplane kt. vzdialenosť k najbližšiemu prvku každého tagu je najväčšia

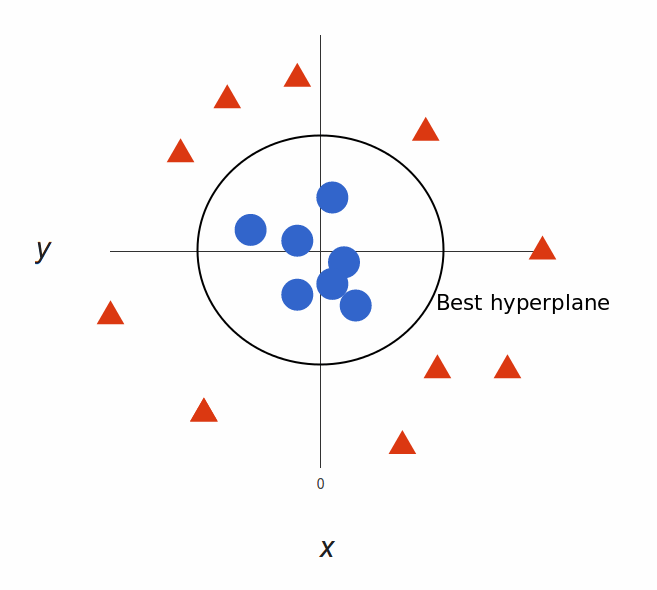
**PRIDANÉ DO DOKUMENTÁCIE**

## Nelineárne dáta

Nie vždy je takéto klasifikovanie také jednoduché. Niekedy môže nastať situácia že máme dáta takto:

V tomto prípade sa to rieši pridaním **ďalšej** (tretej) dimenzie. Po pridaní súradnice **z**priestor bude vyzerať takto:

Po použití SVM v 3d priestore:

To následne premietnuté do 2d priestoru:

## Použitie kernelov

Ako sme zistili nelineárne dáta je možné namapovať s použitím ďalšej dimenzie. Ale výpočet tohto mapovania nie je najjednoduchší, keďže týchto dimenzii môže byť veľa. Pri viacerých datasetoch by to bolo veľmi zložité.

*Here’s a trick: SVM* ***doesn’t need*** *the actual vectors to work its magic, it actually can get by only with the dot products between them. This means that we can sidestep the expensive calculations of the new dimensions! This is what we do instead:*

*Imagine the new space we want:*

*equation*

*Figure out what the dot product in that space looks like:*

**

*Tell SVM to do its thing, but using the new dot product — we call this a kernel function.*

## Porovnanie SVM s kernelmi a bez kernelov

SVM vo viacerých priestoroch je ekvivalent k lineárnemu SVM, lebo takpovediac funguje v 2d priestore. Môžeme sa na to pozerať že ide o mapovanie dát (pravdepodobne nelineárnych) do daného priestoru a potom používanie lineárne SVM.

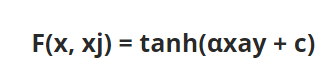
Trik s použitím kernelov funguje tak že namiesto priameho mapovania do použitého priestoru, mapovanie je uskutočnené s použitím *kernelovskej funkcie*.

## Typy kernelov PRIDANE DO DOKUMENTACIE

**Linear kernel**:

* jedná sa o jeden zo základných typov kernelov, zvyčajne v 1d rozmere
* väčšinou používaný v textovo-klasifikačných problémoch (pretože môžu byť lineárne oddelené)
* vzorec: (x, xj sú dáta kt. sa snažíme klasifikovať)

**Sigmoid Kernel**:

* väčšinou používaný pre *neurónové siete*
* tento druh kernela slúži ako dvoj-vrstvový model neurónovej siete, kt. slúži na aktivovanie jednotlivých neurónov
* vzorec:

**Polynomial Kernel**:

* dá sa povedať že sa jedná o generalizovanú reprezentáciu lineárneho kernelu
* nie je až tak používaný, pretože je menej efektívny a menej presný
* vzorec: F(x, xj) reprezentujú *decision boundary* pre rozdelenie danej triedy

# Požitý dataset

Kategorický dataset ktorý určuje že či si daný používateľ zakúpil konkrétny produkt.

Viacej info na: <https://www.kaggle.com/datasets/rakeshrau/social-network-ads>

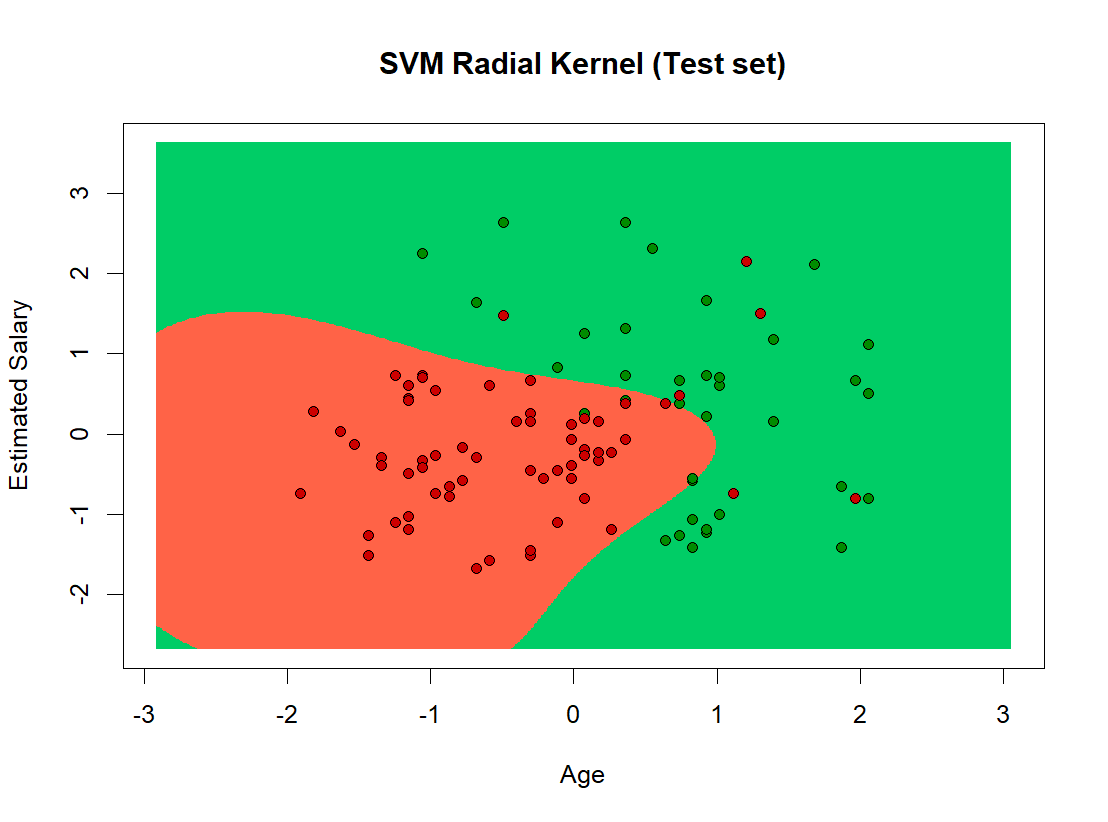
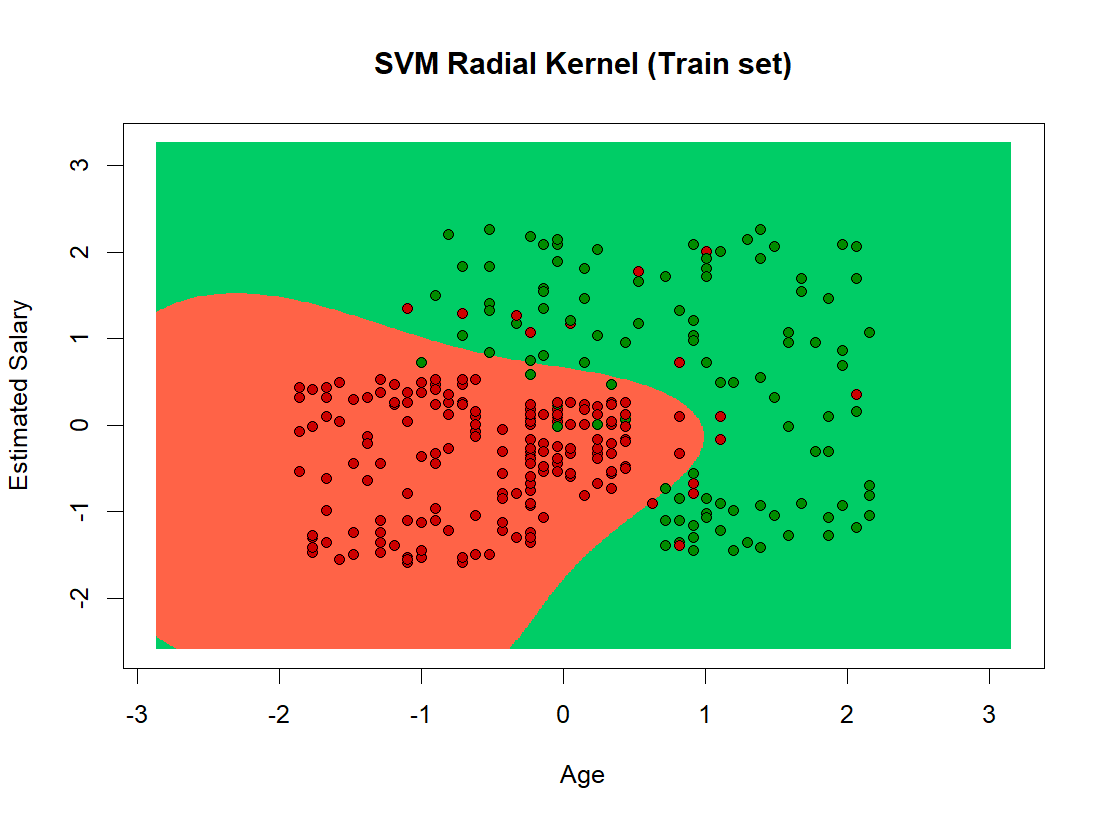
# Postup implementácie vo vybranom jazyku

Postup použitia SVM s kernelmi:

1. Načítanie príslušného datasetu
2. Rozdelenie na trénovaniu a testovaciu množinu
   1. Ak premenné nemajú rovnaké jednotky, použi **scale**()
3. Použitie kernela funkciou **svm**(), kde je potrebné určiť
   1. **Formula**: závislé premenné
   2. **Data**: train alebo test množina
   3. **Type**: '*C-classification'*, pretože chceme použiť regresnú klasifikáciu
   4. **Kernel**: druh použitého kernelu (linear, radial, ...)
4. Predpovedanie výsledkov na trénovacej a testovacej množine pomocou **predict**()

# Výsledky

## Vizuálizácia



## Výsledky