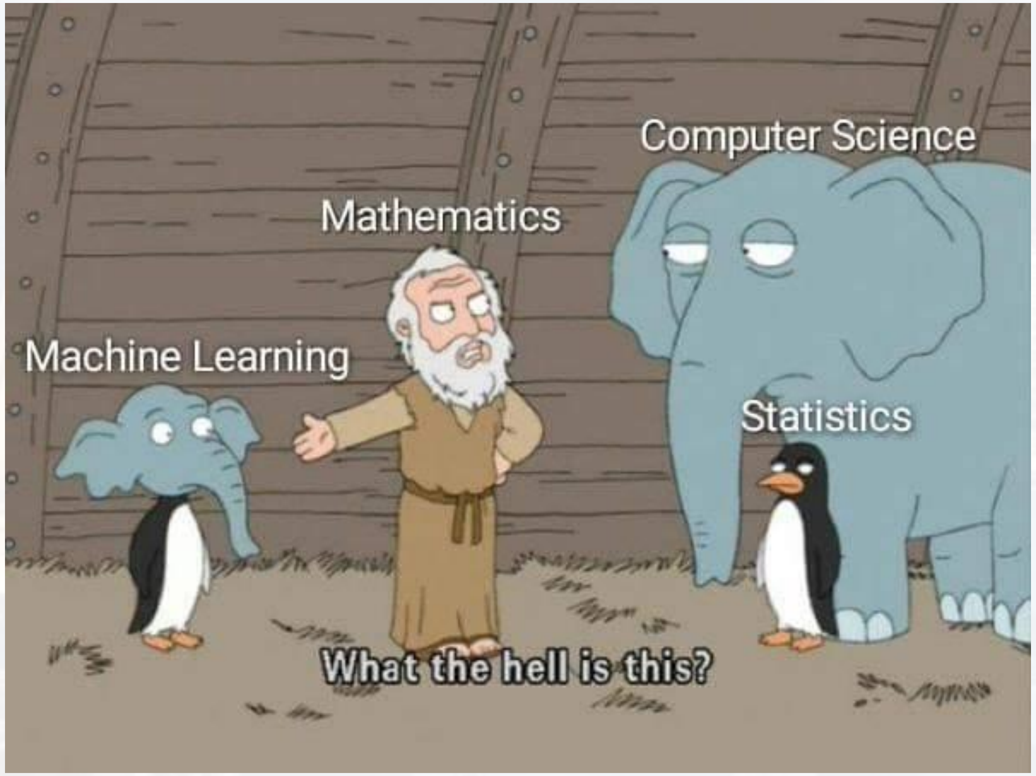
Support Vector Machine

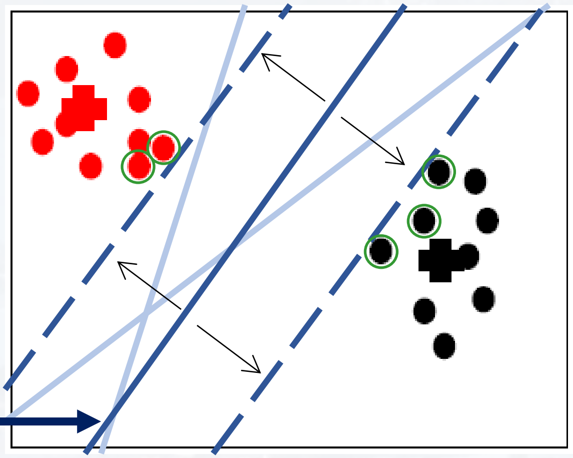


SVM = **S**upport **V**ector **M**achine

idea:

1) finds best **linear** classifier for separating **two** classes by **maximizing margin** using **support vectors**
2) assign new data points to these categories
3) **supervised** learning
4) uses the "kernel trick"

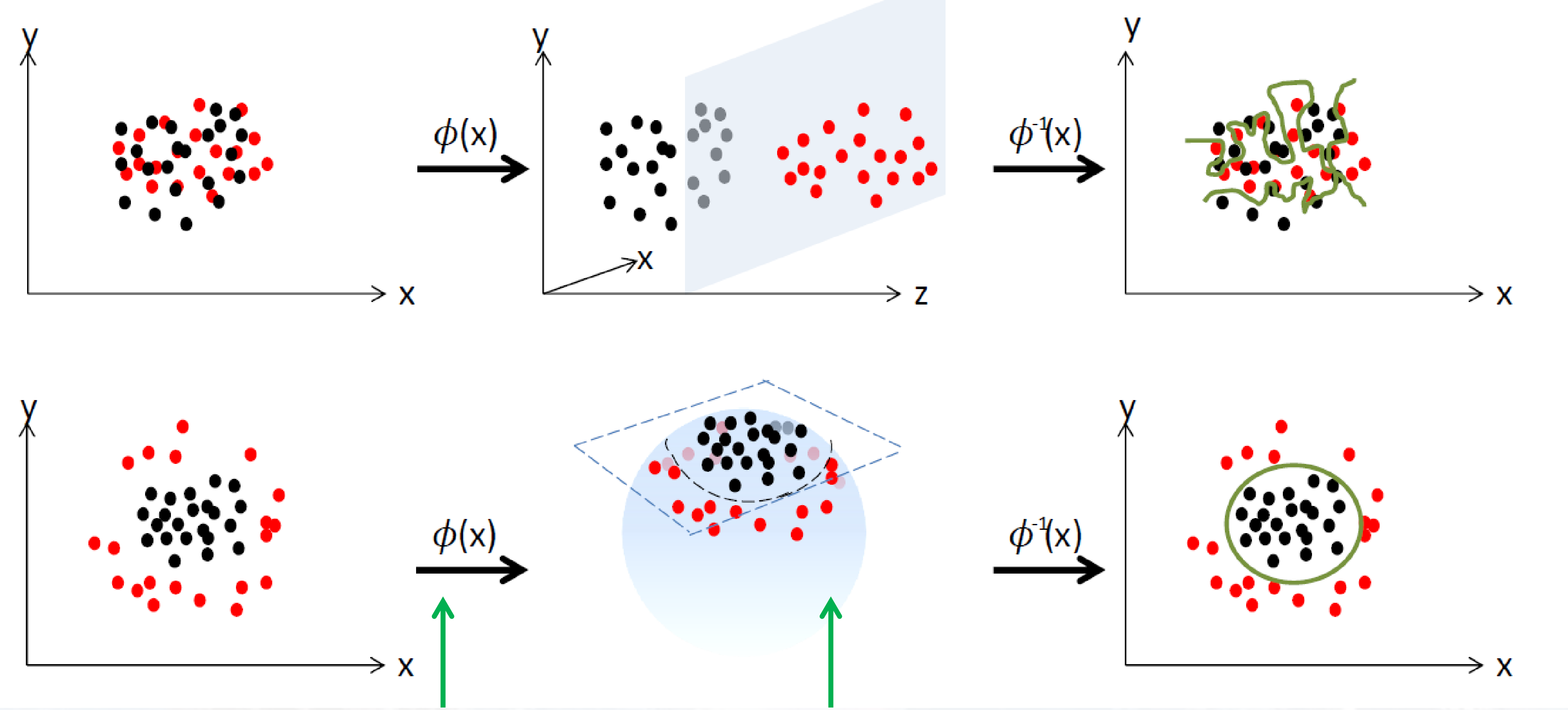
○ support vectors (data points at the edge)



1) What if linear separation is not possible?
2) Is there a multiclass SVM?

1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?

idea: mapping data to higher dimensional feature space



a) data space in N-D

b) a function ϕ maps the data into a M-D (M>N) feature space

c) find hyperplane separator in feature space

d) map back into data space (separator not linear)

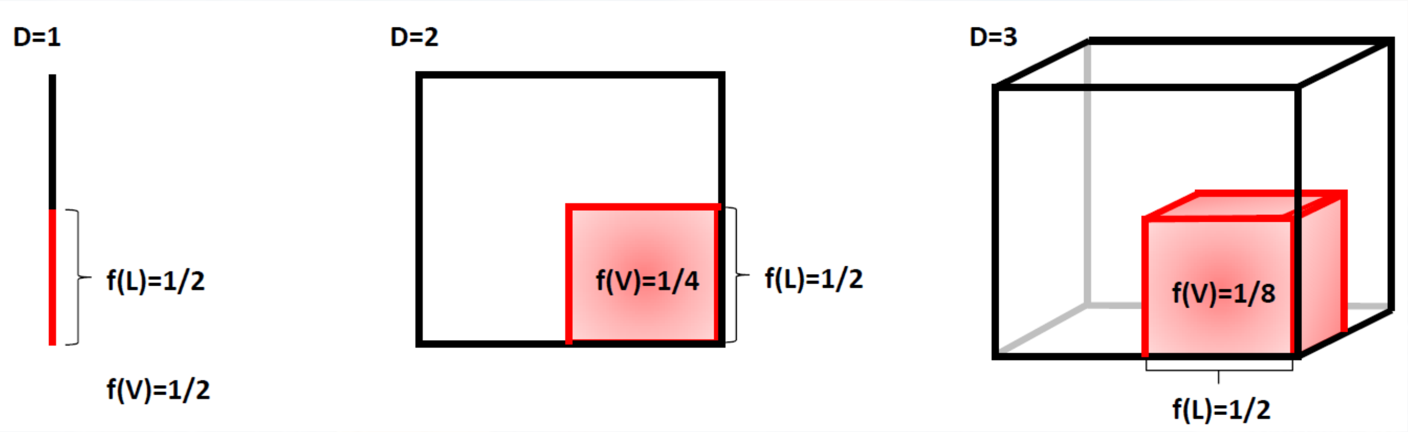
1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?

problems:
- computationally intensive
- ϕ usually unknown

…and dimensionality!!

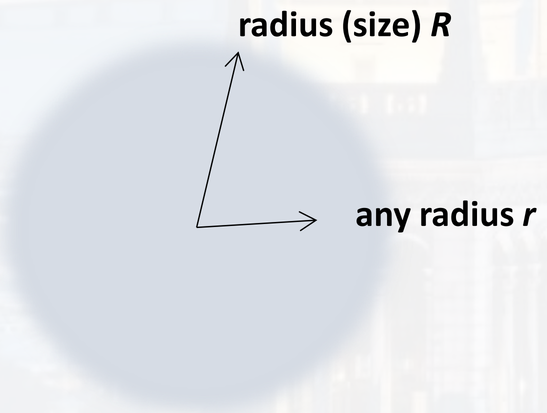
1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?

What is the fraction of volume f(V) covered by a certain fraction of length f(L) for different dimensions D?



answer: $\boldsymbol{f(V) = f(L)^D}$

N - D space



hypersphere:

$$V_D(r) = C(D)\, r^D$$

$C(D)$: constant that only depends on D

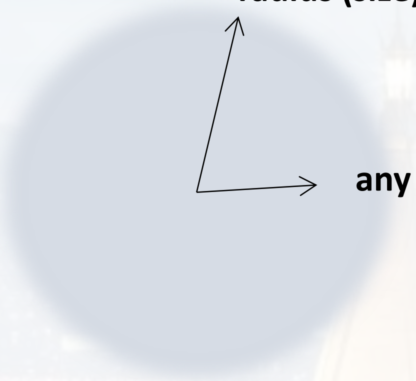
fraction of volume between r and $r - dr$

$$\frac{V_D(r) - V_D(r - dr)}{V_D(r)} = 1 - \frac{(r - dr)^D}{r^D}$$

1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?

N - D space

radius (size) *R*



any radius *r*

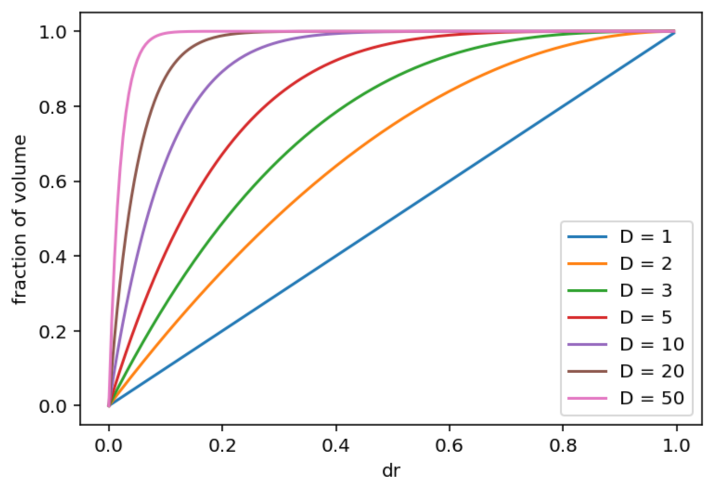
hypersphere:

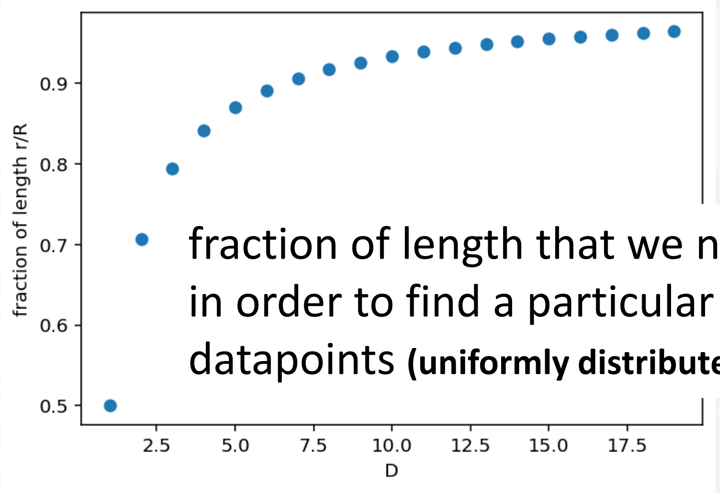
$$V_D(r) = C(D)\, r^D$$

$C(D)$: constant that only depends on D

fraction of volume between r and $r - dr$

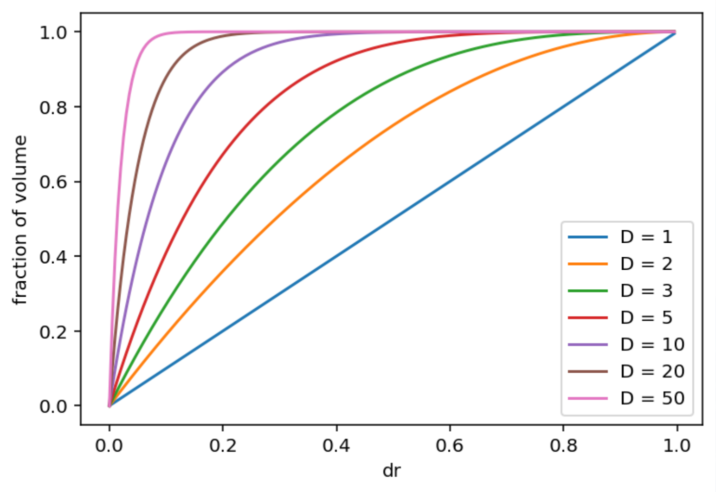
$$\frac{V_D(r) - V_D(r - dr)}{V_D(r)} = 1 - \frac{(r - dr)^D}{r^D}$$

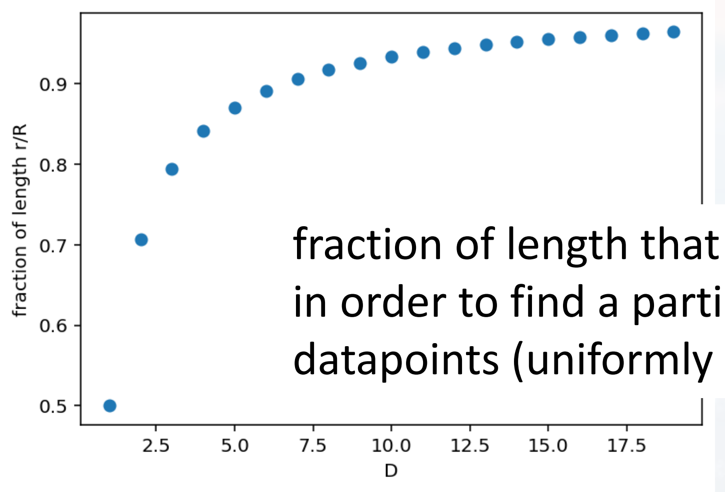




fraction of length that we need to explore in order to find a particular fraction of datapoints **(uniformly distributed in V)**

1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?

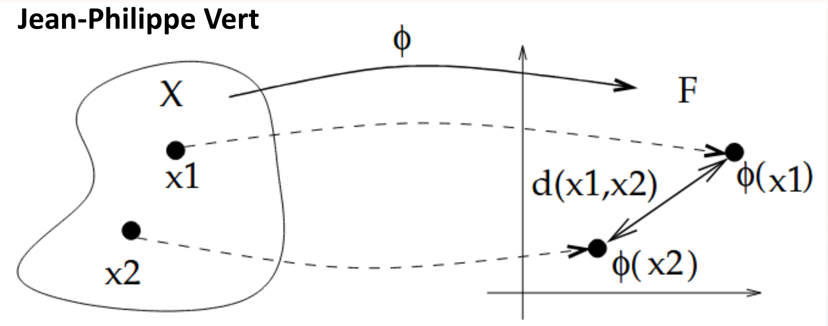




fraction of length that we need to explore in order to find a particular fraction of datapoints (uniformly distributed in R)

- for **large D**, one has to explore a **larger fraction** $\frac{\rho}{R}$ **of the data space** in order to get the **same fraction of data points**
- many algorithms get **less efficient** for large D
- for D→∞, the entire volume is located on the surface of the hyperspace

1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?



problems: - computationally intensive
- ϕ usually unknown

idea: - entire mathematical framework not needed
- for separation: need distances *d* in data space and feature space

$$d^2(x,y) = \langle x-y, x-y\rangle$$

$$d_\phi^2\big(\phi(x),\phi(y)\big) = \langle \phi(x)-\phi(y), \phi(x)-\phi(y)\rangle$$

$$= \langle \phi(x),\phi(x)\rangle - 2\langle \phi(x),\phi(y)\rangle + \langle \phi(y),\phi(y)\rangle$$

kernel $K(x,y) := \langle \phi(x),\phi(y)\rangle$

$$d_\phi^2\big(\phi(x),\phi(y)\big) = K(x,x) - 2K(x,y) + K(y,y)$$

kernel trick: - we don't know *K* either: **we guess it!**

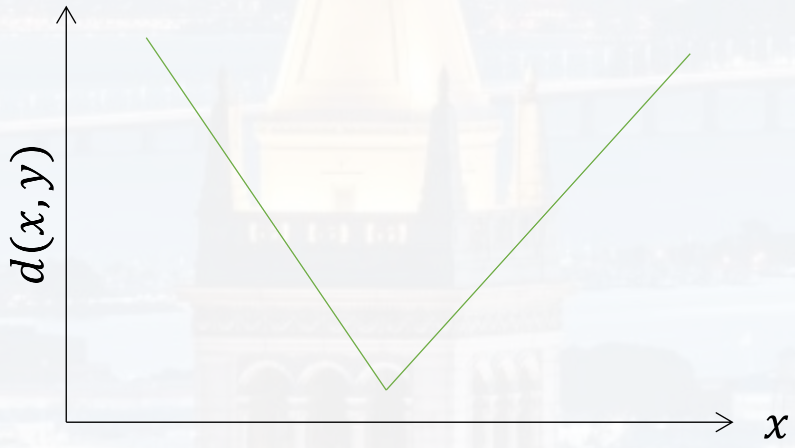
1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?

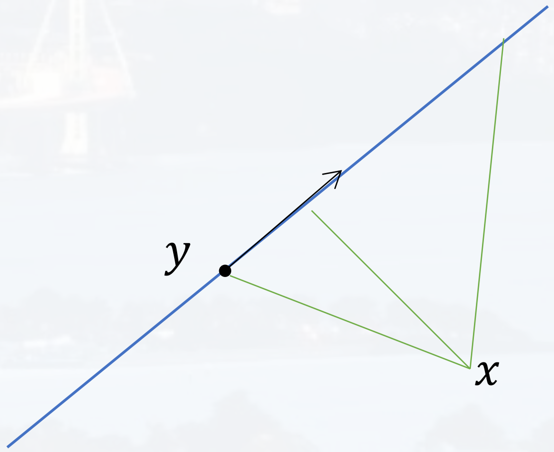
$$d^2\big(\phi(x),\phi(y)\big) = K(x,x) - 2K(x,y) + K(y,y)$$

example: linear kernel $\quad \phi\colon \mathrm{x} \mapsto \mathrm{x} \quad \phi(x) = x$

$$d_\phi^2(x,y) = \langle x,x\rangle - 2\langle x,y\rangle + \langle y,y\rangle = x^2 - 2xy + y^2 = (x-y)^2$$

$$d_\phi^2(x,y) = |x-y|$$

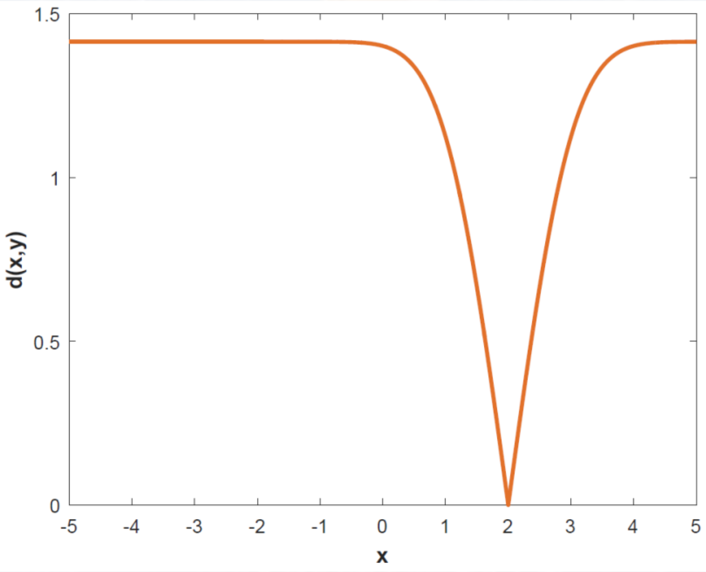


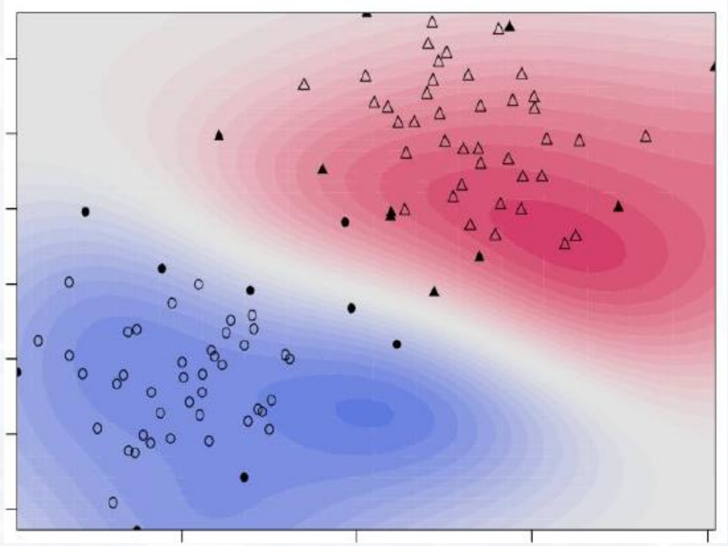


1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?

example: : RBF (**r**adial **b**asis **f**unction) $\phi\colon \mathrm{x} \mapsto \varsigma$ $\phi(x) = \sim\exp(\sim x)$

$$K(x,y) = exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \qquad d_\phi^2(\phi(x),\phi(y)) = 2\left[1 - exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)\right]$$





1) **What if linear separation is not possible?**
2) Is there a multiclass SVM?

kernels available in `sklearn`:

- linear: $K(x, y) = \|x - y\|$

- Gaussian aka RBF (**r**adial **b**asis **f**unction): $K(x, y) = exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$

  in `sklearn` we can adjust $\gamma := \frac{1}{2\sigma^2}$

- polynomial: $K(x, y) = \sum_{n=1}^{N} \|x - y\|^n$

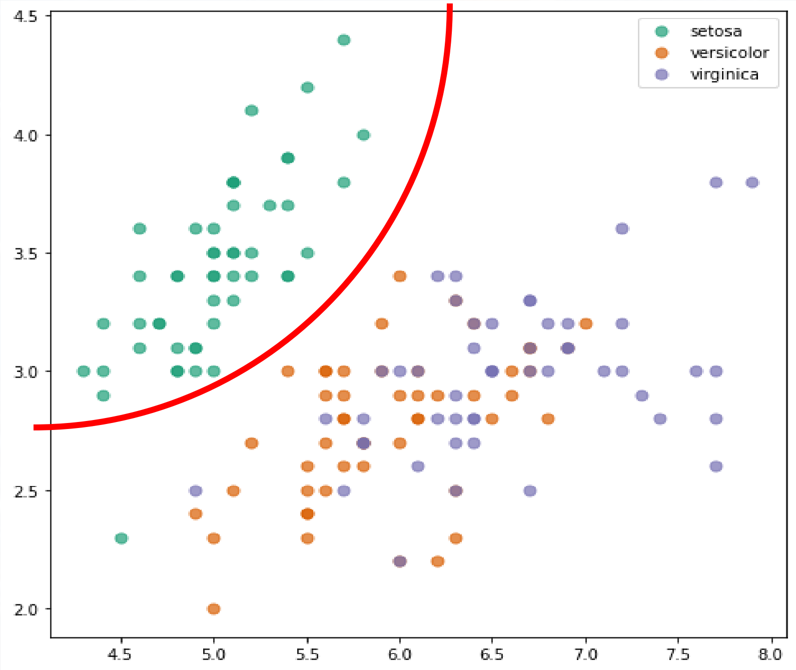
  in `sklearn` we can adjust N

- sigmoidal: $K(x, y) = \frac{e^{\|x-y\|}}{1 + e^{\|x-y\|}}$

1) What if linear separation is not possible?
2) **Is there a multiclass SVM?**

→ one vs rest / one vs one

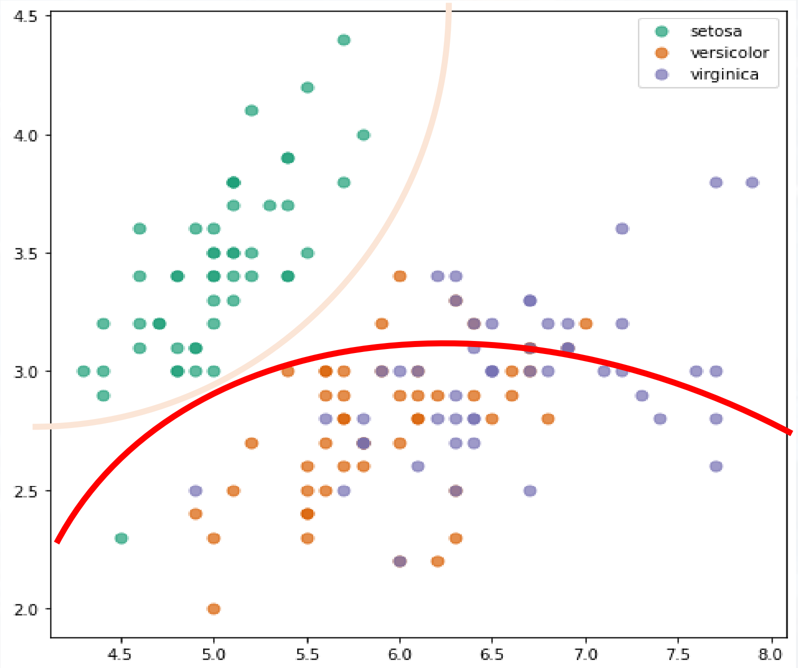
green vs the rest
→ storing probabilities



1) What if linear separation is not possible?
2) **Is there a multiclass SVM?**

→ one vs rest / one vs one

green vs the rest
→ storing probabilities

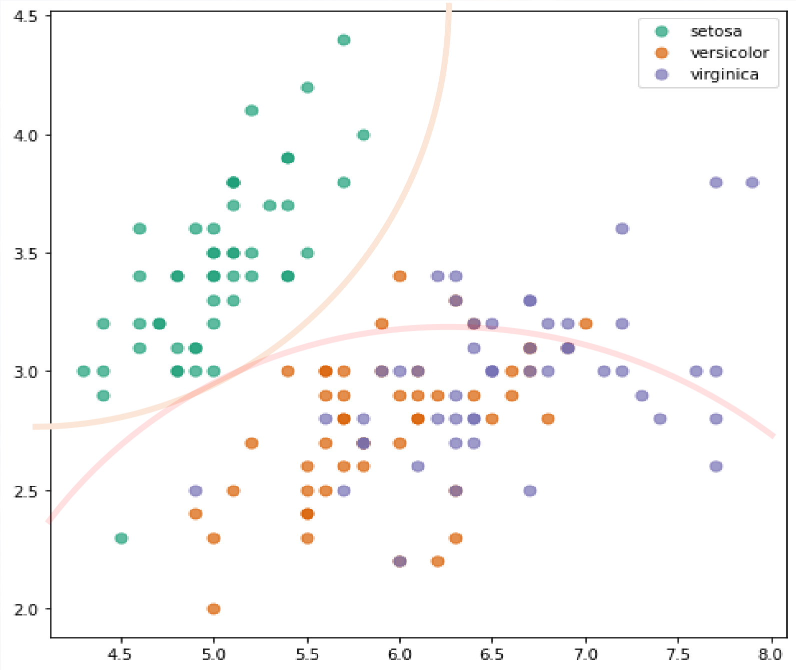


orange vs the rest
→ storing probabilities

1) What if linear separation is not possible?
2) **Is there a multiclass SVM?**

→ one vs rest / one vs one

green vs the rest
→ storing probabilities



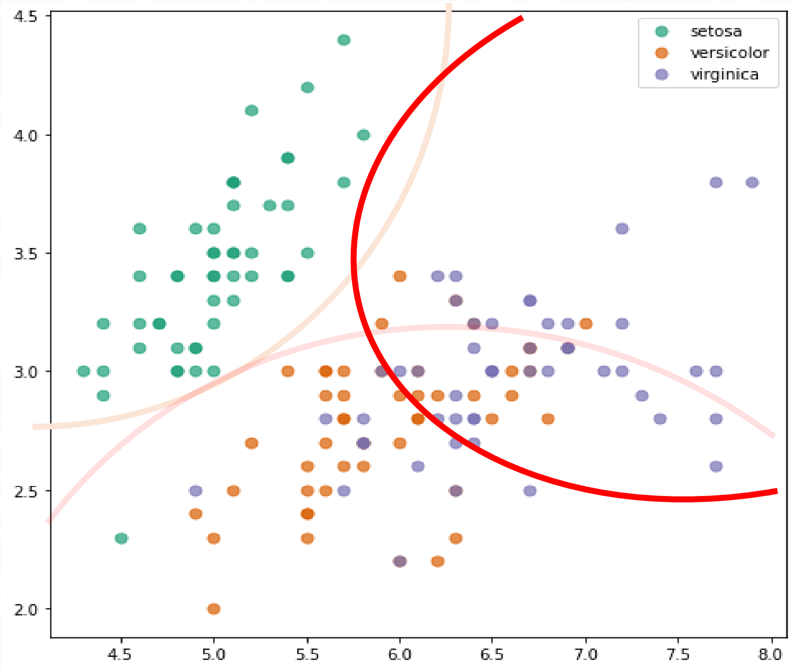
blue vs the rest
→ storing probabilities

orange vs the rest
→ storing probabilities

1) What if linear separation is not possible?
2) **Is there a multiclass SVM?**

→ one vs rest / one vs one

green vs the rest
→ storing probabilities



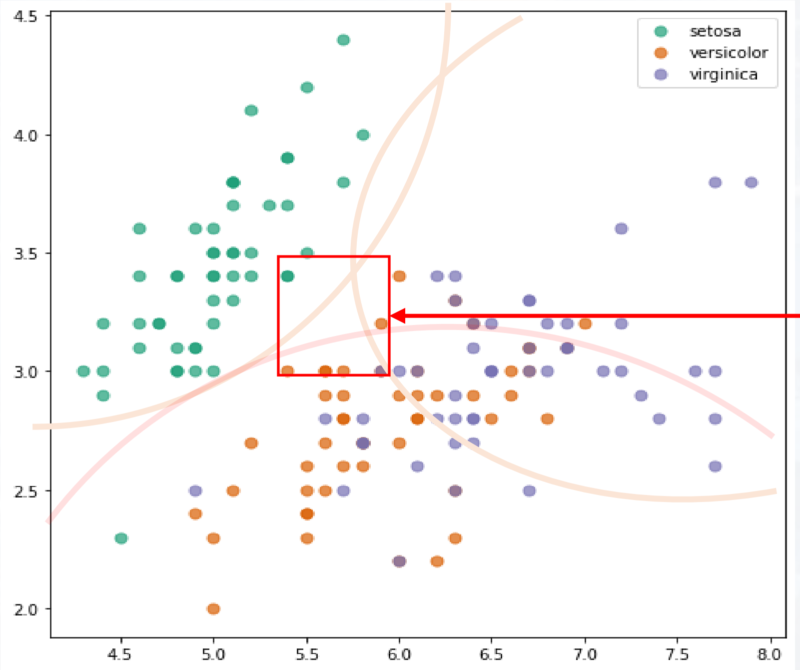
blue vs the rest
→ storing probabilities

orange vs the rest
→ storing probabilities

1) What if linear separation is not possible?
2) **Is there a multiclass SVM?**

→ one vs rest / one vs one

green vs the rest
→ storing probabilities



three probabilities for each data point
→ assign class to most probable value

k class classification with two-class discriminant functions is ambiguous!

```
from sklearn import svm
```

see `Walk_Through_SVM.ipynb`

running analysis with different kernel

1) setting up the model & 2) fitting the model

```
outlinear = svm.SVC(kernel = 'linear', C = 1, decision_function_shape = 'ovr')
linear    = outlinear.fit(X2D, Y)
```

one **v**ersus **r**est

L2 regularization parameter for error tolerance when calculating the classifier

```
outrbf    = svm.SVC(kernel = 'rbf', gamma = 1, C = 1, \
                    decision_function_shape = 'ovr')
rbf       = outrbf.fit(X2D, Y)
```

$\gamma := \frac{1}{2\sigma^2}$

```
outpoly   = svm.SVC(kernel = 'poly', degree = 3, C = 1,\
                    decision_function_shape = 'ovr')
poly      = outpoly.fit(X2D, Y)
```

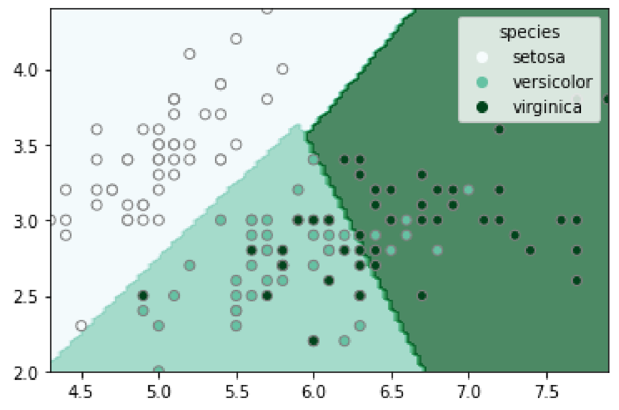
refers to *N* in $\sum_{n=1}^{N} \|x - y\|^n$

```
outsig    = svm.SVC(kernel = 'sigmoid', C = 1, decision_function_shape = 'ovr')
sig       = outsig.fit(X2D, Y)
```

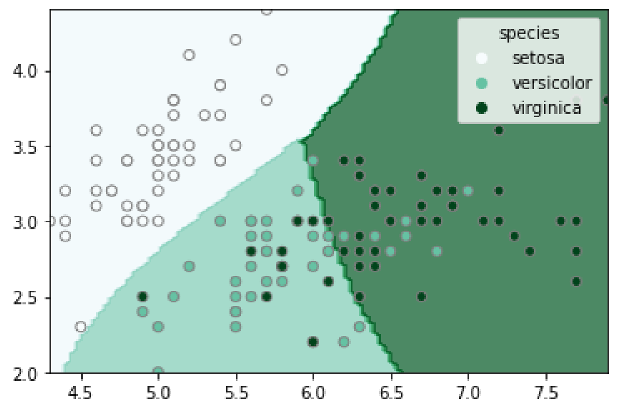
```python
from sklearn import svm
```

see **Walk_Through_SVM.ipynb**

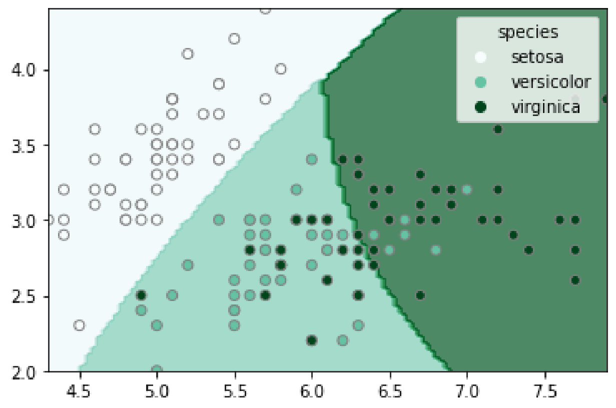
$$K(x, y) = \|x - y\|$$



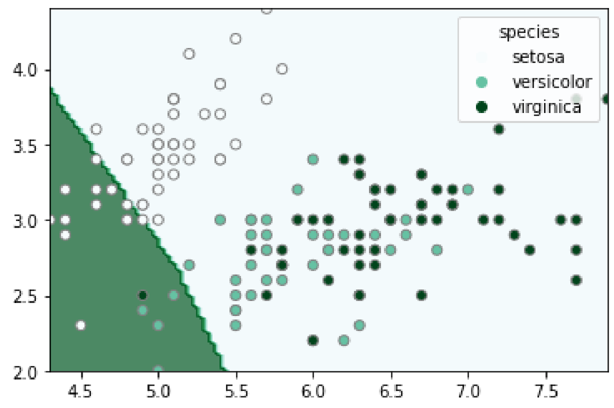
$$K(x, y) = exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$



$$K(x, y) = \sum_{n=1}^{N} \|x - y\|^n$$



$$K(x, y) = \frac{e^{\|x-y\|}}{1 + e^{\|x-y\|}}$$

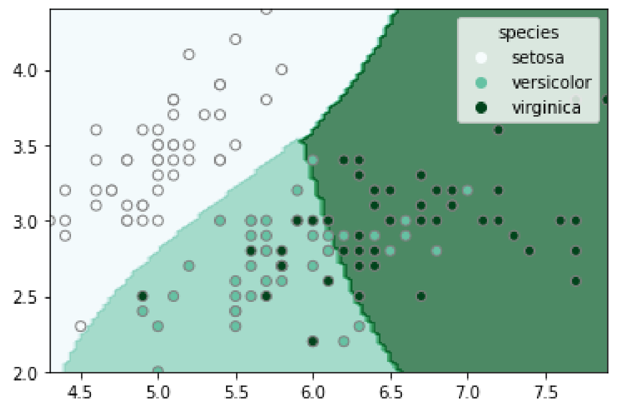


ML Overview

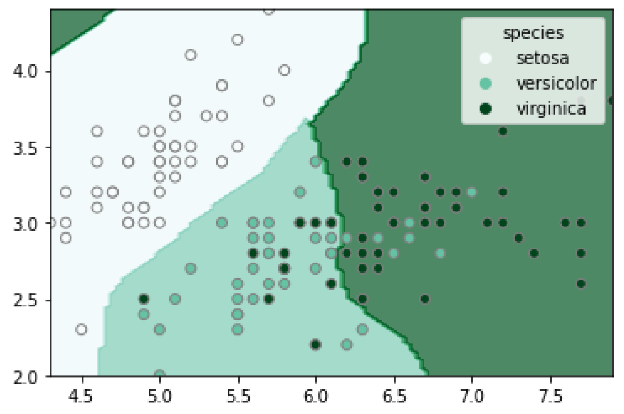

```
from sklearn import svm
```

see Walk_Through_SVM.ipynb

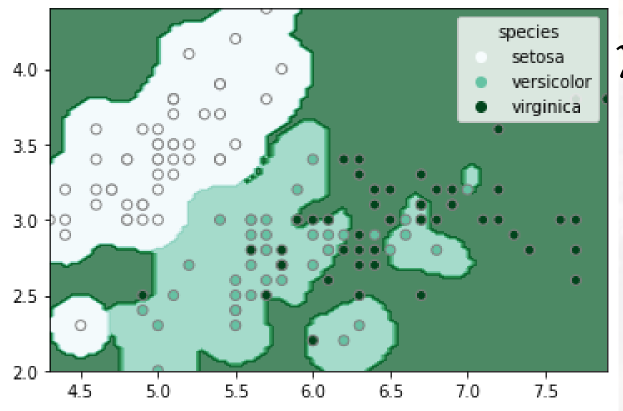
$$K(x,y) = exp\left(-\frac{1}{2\sigma^2}\|x-y\|^2\right)$$



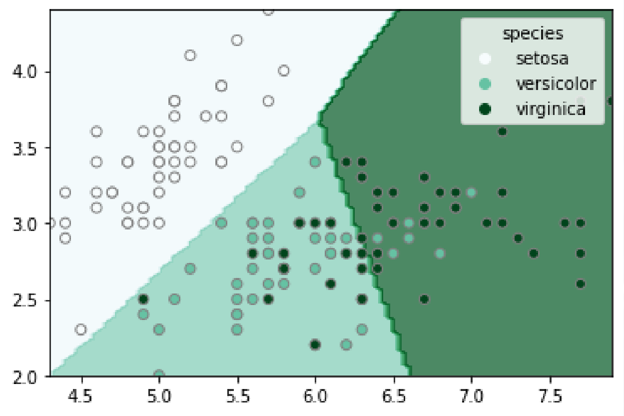
$\gamma := \frac{1}{2\sigma^2} = 1$



$\gamma := \frac{1}{2\sigma^2} = 5$



$\gamma := \frac{1}{2\sigma^2} = 50$

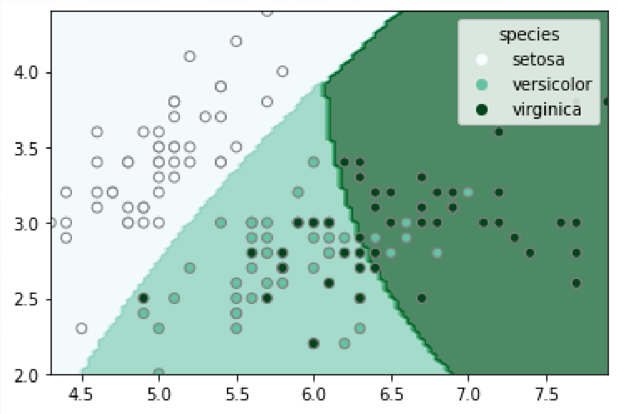


$\gamma := \frac{1}{2\sigma^2} = 0.1$

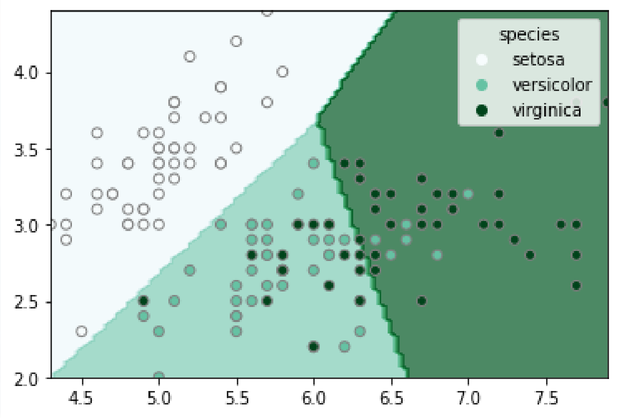
```python
from sklearn import svm
```

see **`Walk_Through_SVM.ipynb`**

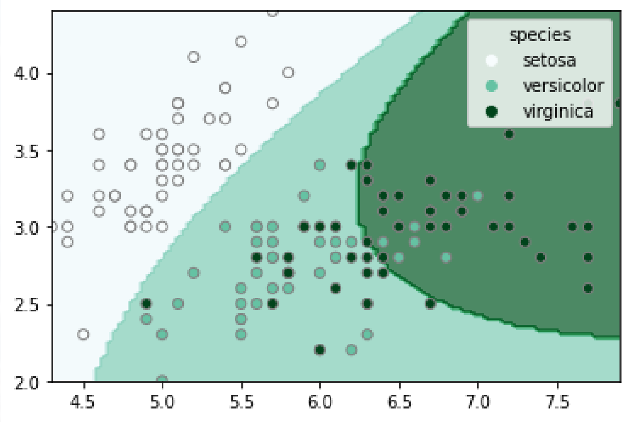
$$\boldsymbol{K(x,y)} = \sum_{n=1}^{N} \|\boldsymbol{x}-\boldsymbol{y}\|^{n}$$



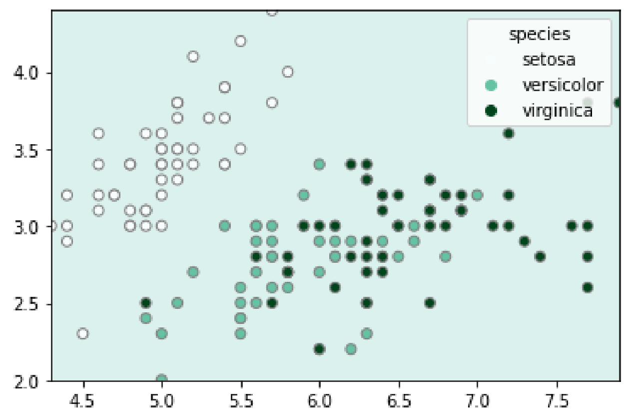
N = 3



N = 1



N = 5



N = 0

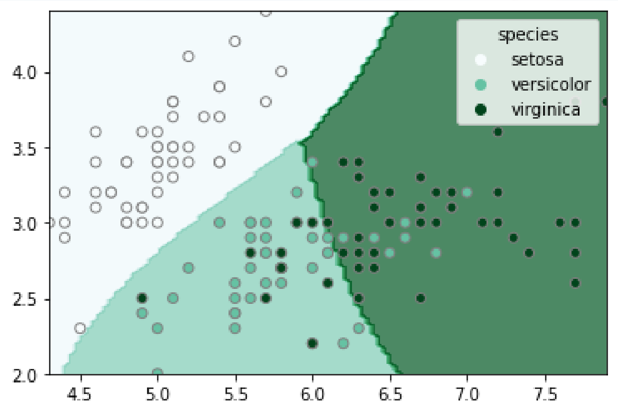
ML Overview


```python
from sklearn import svm
```

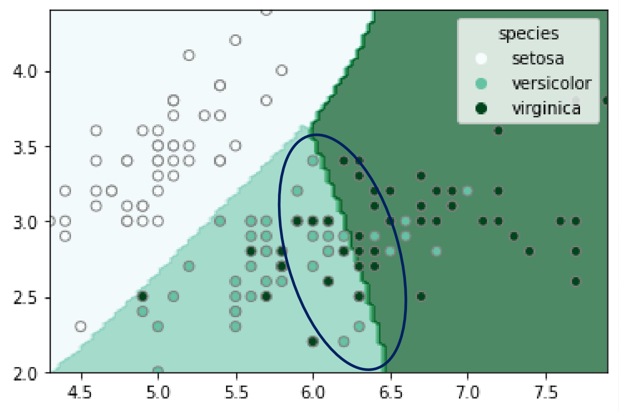
see **`Walk_Through_SVM.ipynb`**

$$K(x, y) = exp\left(-\frac{1}{2\sigma^2}\|x-y\|^2\right)$$

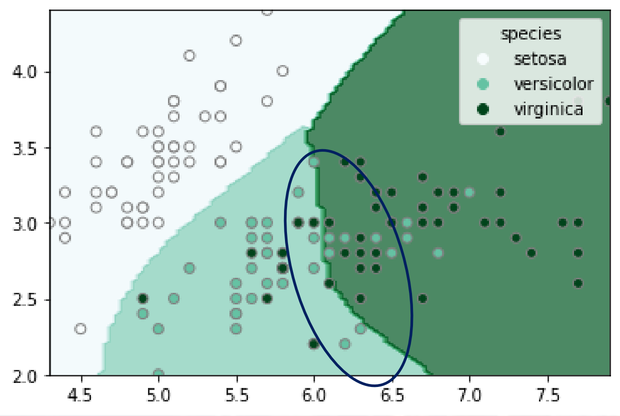
C is a [L2 regularization parameter]()



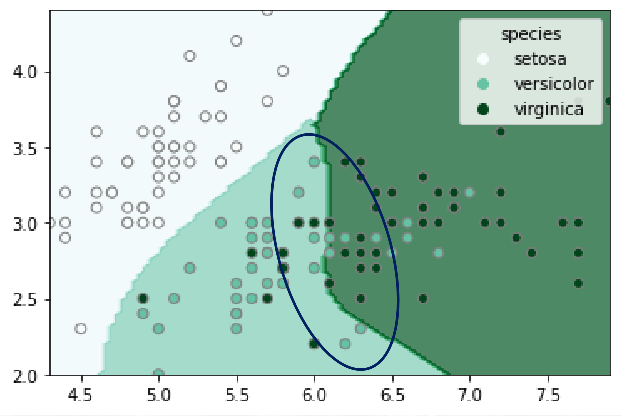
C = 1



C = 0.1



C = 10



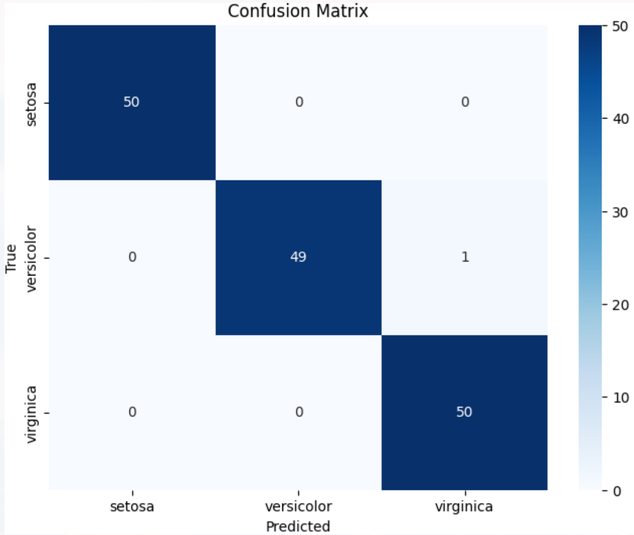
C = 20

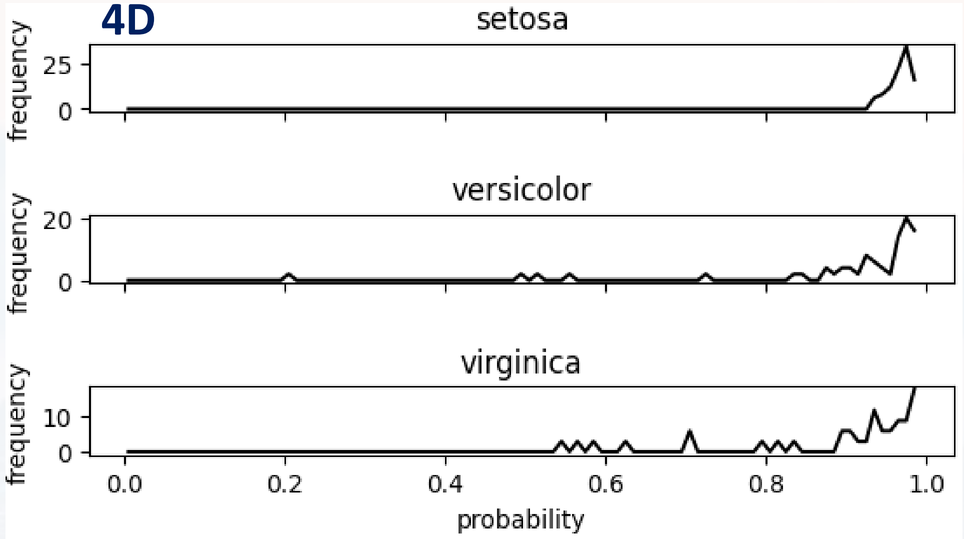
full **4D** data set

3) evaluating the model

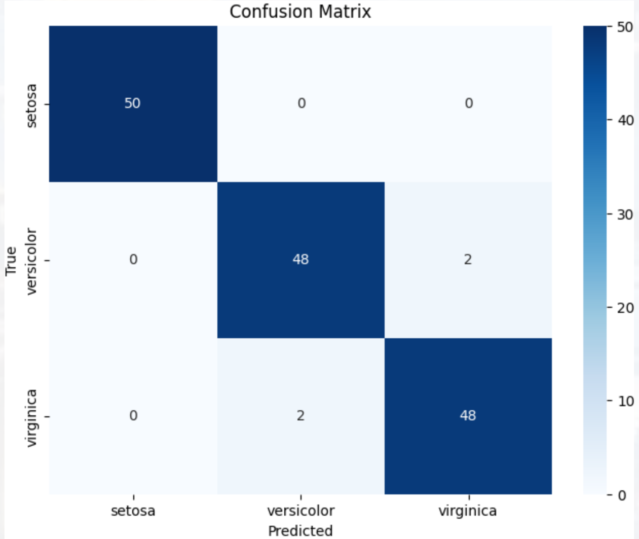
see `Walk_Through_SVM.ipynb`

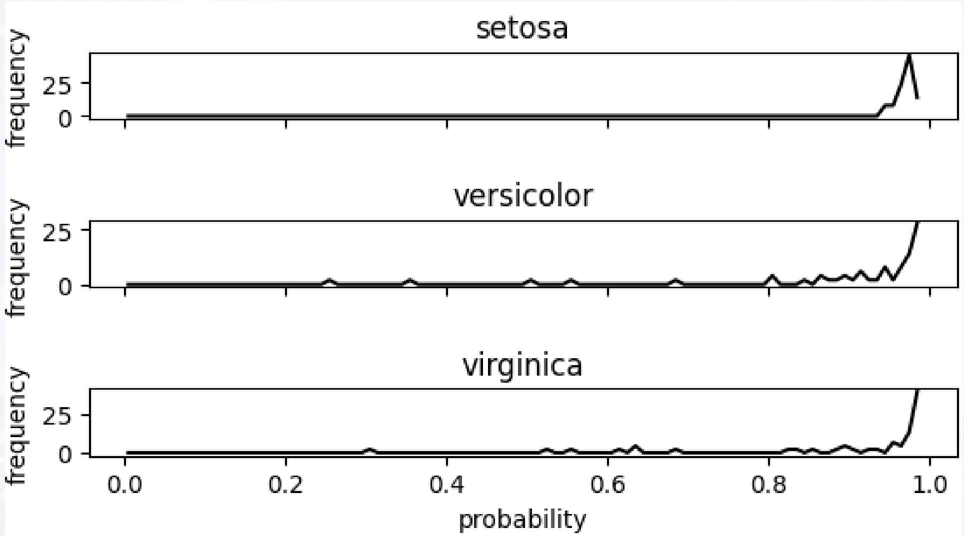
linear
accuracy: 99.3%





Gaussian ($\gamma = 1$)
accuracy: 97.3%



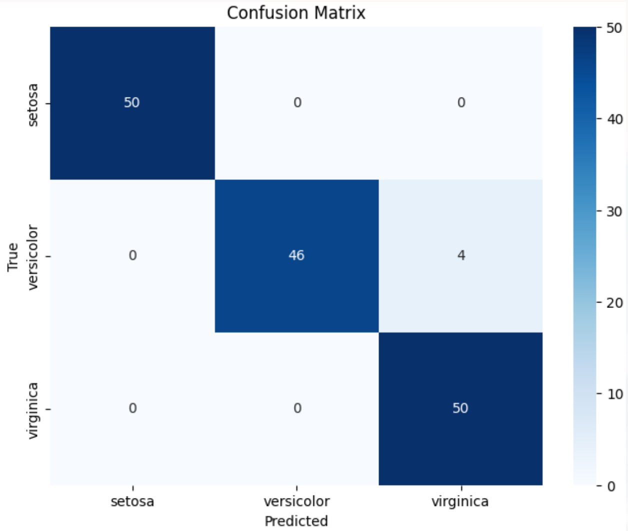


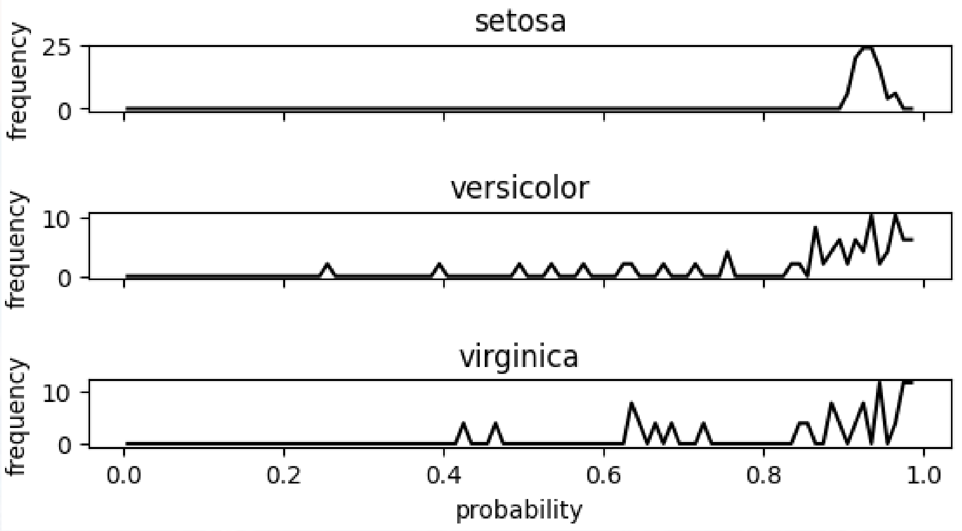
full **4D** data set

3) evaluating the model

see **`Walk_Through_SVM.ipynb`**

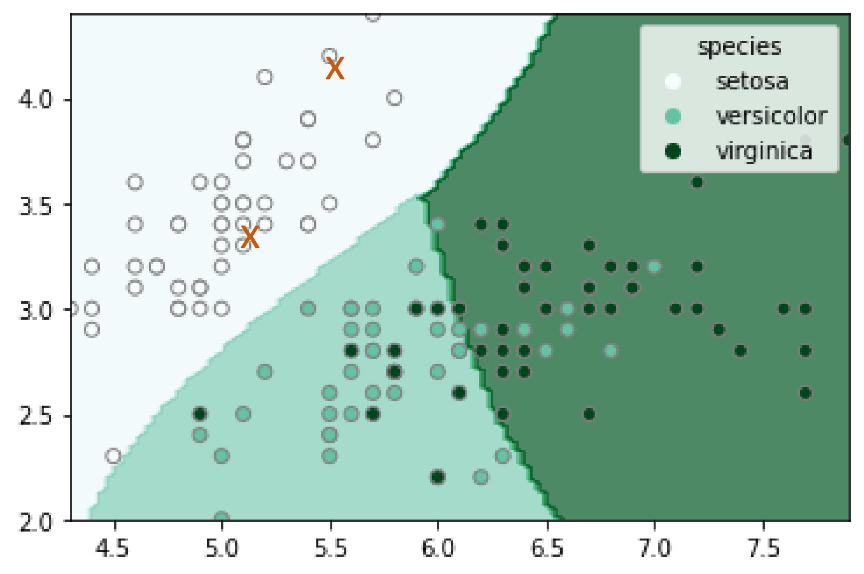
polynomial (n = 3)
accuracy: 97.3%





4) applying the model to a new data set

```
ypred = outpoly.predict([[6, 3.5],[6.3, 4.5]])
```



Thank you very much for your attention!