

Agenda

“Métodos de Aprendizaje Automático”

APRENDIZAJE AUTOMÁTICO

PROGRAMA DE CIENCIA DE DATOS

Profesor: MSc. Felipe Meza

TEC | Tecnológico
de Costa Rica

October 14, 2021



Generalización, Overfitting, Underfitting

- Generalización:
 - Consiste en la capacidad del modelo de aprendizaje de hacer predicciones correctas con el conjunto de pruebas a partir del conjunto de entrenamiento.
 - Si lo hace correctamente se dice que el modelo “generaliza” de conjunto de entrenamiento al conjunto de pruebas.

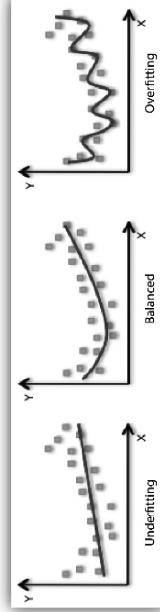
Generalización, Overfitting, Underfitting

- | Age | Number of cars owned | Owns house | Number of children | Marital status | Owns a dog | Bought a boat |
|-----|----------------------|------------|--------------------|----------------|------------|---------------|
| 66 | 1 | yes | 2 | widowed | no | yes |
| 52 | 2 | yes | 3 | married | no | yes |
| 22 | 0 | no | 0 | married | yes | no |
| 25 | 1 | no | 1 | single | no | no |
| 44 | 0 | no | 2 | divorced | yes | no |
| 39 | 1 | yes | 2 | married | yes | no |
| 26 | 1 | no | 2 | single | no | no |
| 40 | 3 | yes | 1 | married | yes | no |
| 53 | 2 | yes | 2 | divorced | no | yes |
| 64 | 2 | yes | 3 | divorced | no | no |
| 58 | 2 | yes | 2 | married | yes | yes |
| 33 | 1 | no | 1 | single | no | no |

 - Podemos experimentar 3 situaciones:
 - Regla 1:** Va comprar un bote el que tenga más de 45 años, rde 3 hijos ó no esté divorciado. [muy complejo]
 - Regla 2:** Va comprar un bote el que tenga una casa. [muy si]
 - Regla 3:** Alguna combinación intermedia.

Generalización, Overfitting, Underfitting

- Overfitting:
 - Escoger un modelo de aprendizaje que sea muy complejo, funciona muy bien con el conjunto de entrenamiento pero no **generaliza** con nuevos datos.
- Underfitting:
 - Escoger un modelo de aprendizaje que sea muy simple, funciona mal con el conjunto de entrenamiento y con nuevos datos no será capaz **generalizar**.



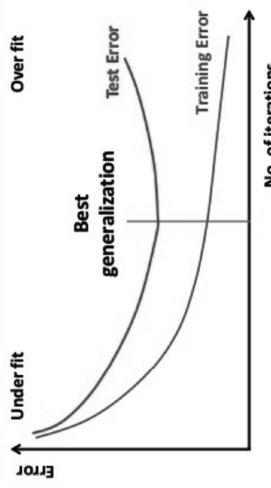
Generalización, Overfitting, Underfitting

- 1
- 2 Training Score: 0.99
- 3 Testing Score: 0.55
- 4
- 5
- 6 Training Score: 0.57
- 7 Testing Score: 0.62
- 8
- 9
- 10 Training Score: 0.82
- 11 Testing Score: 0.86
- 12

Generalización, Overfitting, Underfitting

```
1 # Overfit  
2 Training Score: 0.99  
3 Testing Score: 0.55  
4  
5 # Underfit  
6 Training Score: 0.57  
7 Testing Score: 0.62  
8  
9 # Fit  
10 Training Score: 0.82  
11 Testing Score: 0.86
```

Generalización, Overfitting, Underfitting



Regresión Lineal Ordinaria (OLS)

- Consiste en encontrar los parámetros w y b que minimicen el MSE (mean square error) entre los valores reales y la predicción.
 - En la mayoría de herramientas computacionales se le conoce como “Linear Regression”.

```
from sklearn.linear_model import LinearRegression
```

卷之三

```
5 lr.score(X_train, y_train)
```

II : ACCORDS (A-C-G-E-N, J-GEN)

三

$$\min_{w,b} \frac{1}{N} \sum_{i=1}^N (f_{w,b}(x_i) - y_i)^2$$

1

$$\min_{w,b} C|w| + \frac{1}{N} \sum_{i=1}^N (f_{w,b}(x_i) - y_i)^2, \text{ donde } |w| \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^D |w^{(i)}|^2}$$

$\min_{w,b} C\|w\|^2 + \frac{1}{N}\sum_{i=1}^N (f_{w,b}(x_i) - y_i)^2$, donde $\|w\|^2 = \sum_{j=1}^B (w^{(j)})^2$

Regresión Lineal RIDGE y LASSO

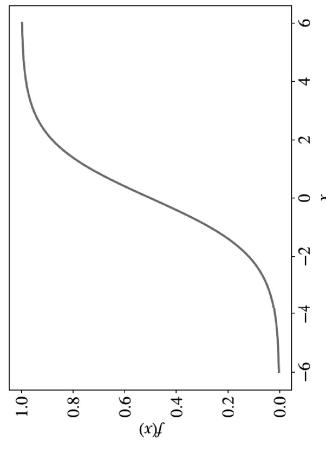
Regresión Lineal RIDGE y LASSO

- LASSO (L1)
 - Se pueden obtener valores de w **iguales** a cero.
 - Reduce la **complejidad** y el **número de variables**.
 - RIDGE (L2)
 - Modelo que persigue obtener valores de w **cercanos** al cero con el fin de minimizar el efecto de los atributos mientras se logra un buen desempeño.
 - El parámetro α permite balancear la simplicidad y el desempeño. Su valor óptimo depende de las características del conjunto de datos.
 - Reduce la **complejidad** pero no el número de variables.

Modelos Lineales

Logistic Regression

- Se busca una función que logre operar como una probabilidad de la forma $0 \leq p \leq 1$, esa función es la **logística** o **sigmoid**:



$$f(x) = \frac{1}{1+e^{-x}}$$

Logistic Regression

- El modelo queda de la siguiente forma:

$$f_{w,b}(x) \stackrel{def}{=} \frac{1}{1+e^{-(wx+b)}}$$

- El criterio consiste en maximizar la probabilidad de los datos de entrenamiento de acuerdo al modelo:
$$L_{w,b} \stackrel{def}{=} \prod_{i=1,\dots,n} f_{w,b}(x_i)^{y_i} (1 - f_{w,b}(x_i))^{(1-y_i)}$$
 - Con el fin de facilitar el uso de la ecuación anterior, se recurre al uso de logaritmos.

$$\log L_{w,b} \stackrel{def}{=} \ln(L_{(w,b)}(x)) = \sum_{i=1}^N y_i \ln f_{w,b}(x) + (1 - y_i) \ln(1 - f_{w,b}(x))$$

SVM (Máquinas de Vector de Soporte)

Se debe cumplir que:

$$wx_i - b \geq 1 \text{ if } y_i = +1, \\ wx_i - b \leq -1 \text{ if } y_i = -1$$

Distancia (euclíadiana) a maximizar:

$$\sqrt{\sum_{j=1}^D (w(j))^2}$$

Criterio de optimización:

$$\min |w| = \min \frac{1}{2} \|w\|^2$$

卷之三

卷之三

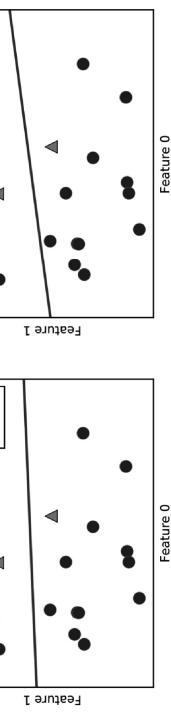
卷之三

SVM (Máquinas de Vector de Soporte)

- Ventajas:
 - Buena precisión.
 - Opera bien con datos pequeños/limpios.
 - Desventajas:
 - Cuando el conjunto de entrenamiento es muy grande, se puede volver lento.
 - No dà buenos resultados cuando los datos son “ruidosos”.

Logistic Regression y SVM

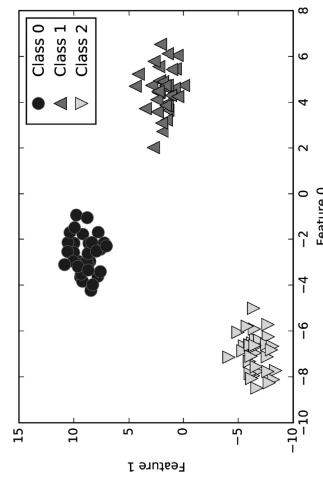
- Se utiliza el parámetro **C** para controlar la regularización.
- Un valor alto de **C** implica menos regularización, el modelo trata de ajustarse a los datos de training lo mejor posible.
- Un valor bajo de **C** implica encontrar coeficientes w cercanos a cero.



Logistic Regression y SVM

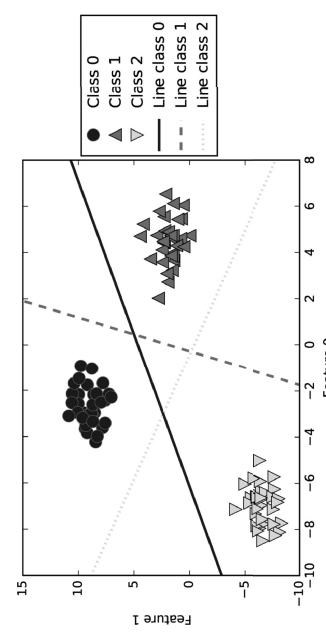
Modelos Lineales Multi-Clase

- Consiste en usar modelos lineales usados comúnmente en clasificación binaria a clasificación con múltiples clases.



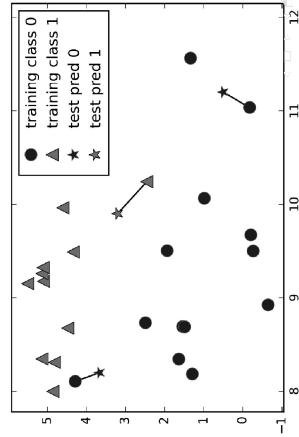
Modelos Lineales Multi-Clase

- Se recurre al algoritmo *uno vs. el resto*, el cual consiste en separar cada clase del resto como si fuera clasificación binaria.



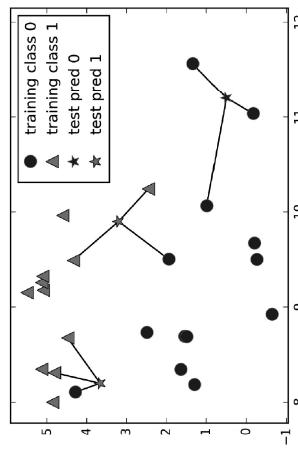
kNN - k Nearest Neighbors (Clasificación)

- Es uno de los más **simples** algoritmos de aprendizaje, en cuanto a criterio de predicción.
- Para predecir un nuevo dato, busca el /los punto(s) más cercanos i.e vecino(s) más cercano(s).
- El caso más elemental consiste en hacer el cálculo a partir de 1 vecino cercano:



kNN - k Nearest Neighbors (Clasificación)

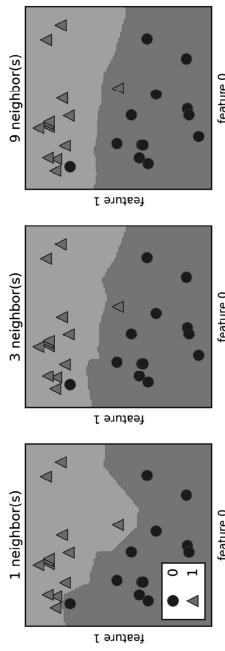
- En caso de considerar más de 1 vecino se usa la regla del "voto" i.e un conteo de la cantidad de vecinos cercanos pertenecientes a una clase, veamos el caso de $k = 3$.



- Este método puede ser aplicado a conjuntos de datos de múltiples clases.

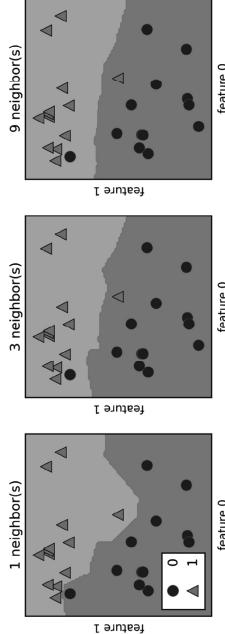
kNN - k Nearest Neighbors (Clasificación)

- El **Límite de decisión** se define como la línea que separa a una clase de otra.



kNN - k Nearest Neighbors (Clasificación)

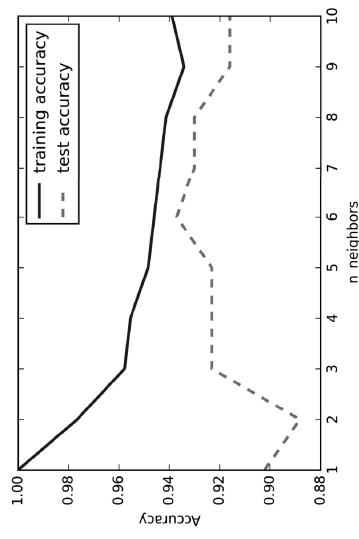
- El **Límite de decisión** se define como la línea que separa a una clase de otra.



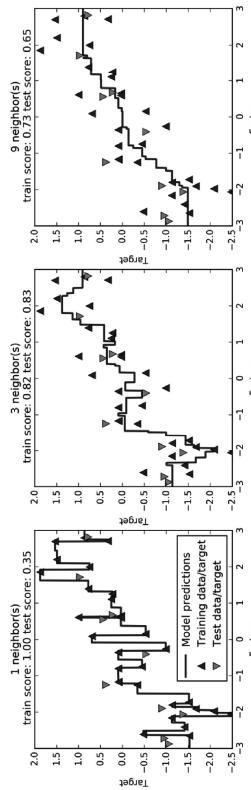
- En este caso observamos como un **k** bajo representa un modelo de alta complejidad, y un **k** alto un modelo de baja complejidad.
- Al ser **k** un valor del cual tenemos control, se le conoce como **hiper-parámetro**.

kNN - k Nearest Neighbors (Clasificación)

- Se trata de encontrar el punto en el cual: el conjunto de pruebas entregue la mejor precisión de la mano de una precisión del conjunto de entrenamiento "buena" (i.e no perfecta pero tampoco pésima).

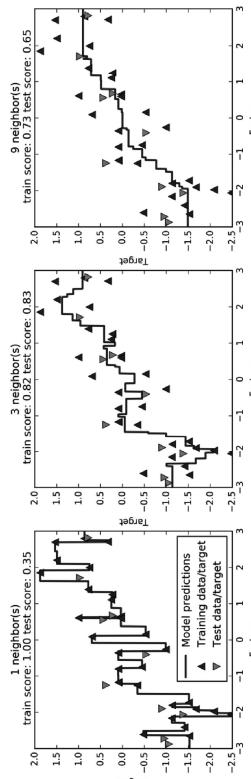


kNN - k Nearest Neighbors (Regresión)



De nuevo, nótense como un k bajo representa un modelo de alta complejidad, y un k alto un modelo de baja complejidad.

kNN - k Nearest Neighbors (Regresión)



De nuevo, nótense como un k bajo representa un modelo de alta complejidad, y un k alto un modelo de baja complejidad.

kNN - k Nearest Neighbors

- Hiper-Parámetros:
 - Requiere de ajuste del valor **k** y de la forma de medir la **distancia**, se asumirá la distancia euclíadiana para efectos prácticos.
- Ventajas:
 - Muy fácil de entender.
 - Generalmente entrega buenos resultados sin excesivo ajuste.
- Desventajas:
 - Cuando el conjunto de entrenamiento es muy grande, se puede volver lento.
 - No dà buenos resultados cuando hay *muchos "features"* o muchos son ceros.



naive Bayes

- El teorema de Bayes establece que:
 - $$P(A|B) = \frac{P(A) \times P(B|A)}{P(B)}$$
 donde A es la hipótesis, $P(A)$ la probabilidad a priori, $P(B|A)$ la probabilidad condicional, $P(B)$ la probabilidad marginal y $P(A|B)$ la probabilidad a posteriori.
- Por ejemplo, se va fabricar una tarjeta electrónica y se requiere de un componente B , las partes se compran a 2 proveedores distintos.
 - Un 30% se le compra al proveedor A_1 y un 70% al proveedor A_2 .
 - Históricamente el 6% de las partes compradas al proveedor A_1 son defectuosas, en el caso del proveedor A_2 solo el 4% son defectuosas.



naive Bayes

- Se tiene entonces que:
 - $P(A_1) = 0.3$
 - $P(A_2) = 0.7$
 - $P(B|A_1) = 0.06$
 - $P(B|A_2) = 0.04$
 - $P(B') = 60 \rightarrow 20$

$$P(A_1|B) = \frac{P(A_1) \times P(B|A_1)}{P(B)} = \frac{0.3 \times 0.06}{0.046} = 0.39$$

$$P(A_2|B) = \frac{P(A_2) \times P(B|A_2)}{P(B)} = \frac{0.7 \times 0.04}{0.046} = 0.61$$

- tiene entonces que:

 - $P(A_1) = 0.3$
 - $P(A_2) = 0.7$
 - $P(B|A_1) = 0.06$
 - $P(B|A_2) = 0.04$
 - $D'(D) = \frac{60}{100} \cdot 40 \cdot 20 = \mathbf{1\ 60'}$
 - $D(D) = \frac{100}{100} \cdot 40 \cdot 70 = \mathbf{2\ 80'}$
 - $D''(D) = \frac{100}{100} \cdot 40 \cdot 70 = \mathbf{1\ 60'}$

- Se dice “naive” por el hecho de que supone una idea simple: independencia de los atributos entre sí.
 - Se puede encontrar en 3 tipos, dependiendo de las características de los datos:
 - BernoulliNB: Datos son binarios.
 - MultinomialNB: Datos representan un conteo.
 - GaussianNB: Distribución gaussiana de los datos.
 - Es eficiente por que aprende a través de la recolección de estadísticas por clase de cada atributo.
 - MultinomialNB: Valor promedio de cada atributo para cada clase.
 - GaussianNB: Valor promedio de cada atributo y STD para cada clase.

naive Bayes

- Algoritmo suele ser muy rápido en el entrenamiento y predicción.
- Se suele usar mucho en grandes conjuntos de datos.

- MultinomialNB y GaussianNB, usan el parámetro α para suavizado de la generalización, resultando en modelos menos complejos.

```
1 from sklearn.naive_bayes import MultinomialNB  
2 clf = MultinomialNB(alpha=1.0)  
3 clf.fit(X, Y)
```

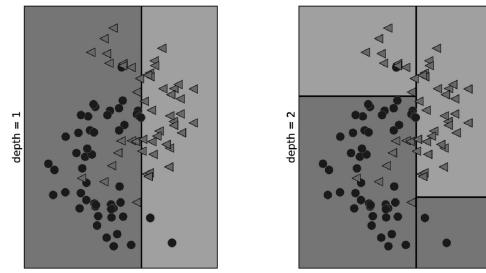
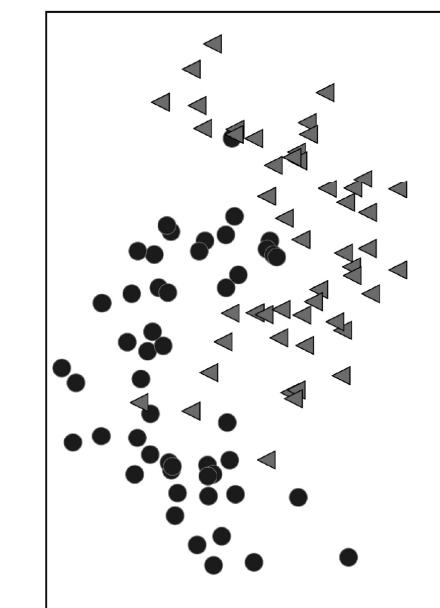
- GaussianNB se usa con datos de grandes dimensiones, mientras MultinomialNB y BernoulliNB con datos dispersos.

- Esquema de aprendizaje basado en preguntas tipo if/else, seguidas de una decisión.
- A cada cuadro, sea una pregunta o nodo terminal, se le llama hoja.
- A cada pregunta se le conoce como prueba.
- Al primer nodo se le llama raíz y es el que parte los datos de la manera más significativa.



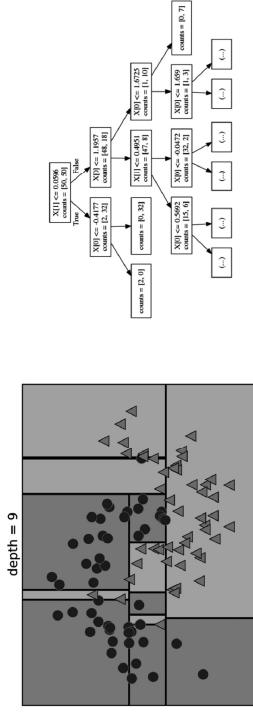
Decision Trees

Decision Trees



Decision Trees

Decision Trees



La hoja cuyos datos pertenecen solamente a una clase se llama *pura*.

Queremos llegar a alcanzar un DT con sólo hojas puras?

- Un árbol de decisión con sólo hojas puras implica que está 100% ajustado al conjunto de entrenamiento, en decir: **overfitting**.
 - Existen dos técnicas para evitarlo:
 - **pre-pruning:** Detener la creación del árbol tempranamente (e.g limitar máximo de hojas o la profundidad del árbol).
 - **pruning (post-pruning):** Remover o colapsar nodos que contienen poca información.
 - Se puede tener una idea de la importancia de los atributos en el árbol usando el comando:

tree feature importance

- O bien, mediante algún método de interpretación gráfica.

Decision Trees

- Los árboles de decisión pueden ser usados para clasificación o regresión:
- `DecisionTreeClassifier`
- `DecisionTreeRegressor`
- Es muy recomendado el uso de técnicas para evitar overfitting:
 - `max_depth`
 - `max_leaf_nodes`
 - `min_samples_leaf`
- Son modelos fáciles de visualizar.
- No requieren pre-procesado en exceso por que se desempeña bien con datos en múltiples escalas.

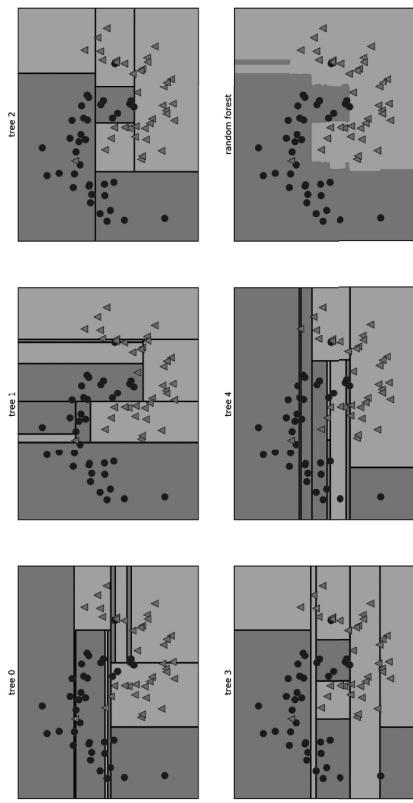
Random Forest

- Algoritmo que pertenece a la familia de métodos en "conjunto" donde se recurre al uso de múltiples modelos de aprendizaje automático operando como uno sólo.
- Es una propuesta para evitar caer en "overfitting" usando árboles de decisión convencionales.
- Consiste en una colección de árboles de decisión, donde cada árbol es diferente.
- En conjunto se asume algún overfitting para cada árbol, sin embargo al promediar los resultados se reduce el overfitting.
- Su nombre proviene de la forma en la cual se construye cada árbol para mantenerlos diferentes, es una forma aleatoria.

Random Forest

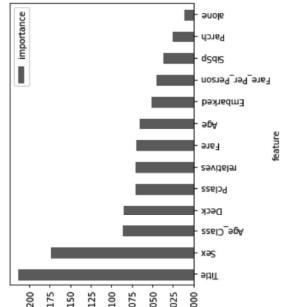
- Para construir el árbol, se debe indicar el número de árboles mediante el parámetro `n_estimators`.
 - Es posible usar el método tanto para la clasificación como para la regresión:
 - `RandomForestRegressor`
 - `RandomForestClassifier`
 - El parámetro `max_features` define que tan diferentes son los árboles que se crearán. Un valor alto implica similitud en los árboles uno bajo significa amplia diferencia.
 - Para la predicción usando Random Forest, primero se lleva a cabo la predicción para cada árbol y luego si es clasificación se hace un conteo del resultado de cada árbol (gana la mayoría) y si es regresión se estima el promedio.

Random Forest

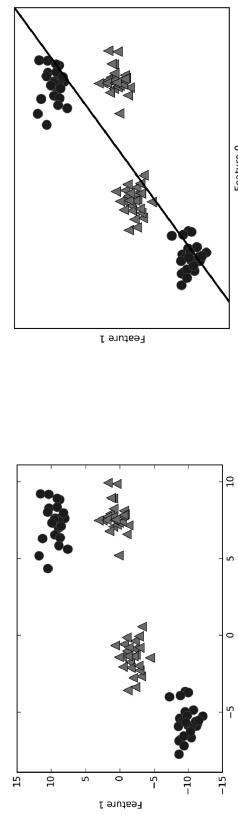


Random Forest

feature	importance
Title	0.213
Sex	0.173
Age_Class	0.086
Deck	0.085
Pclass	0.071
relatives	0.070
Fare	0.069
Age	0.065
Embarked	0.051
Fare_Per_Person	0.045
SlbSp	0.037
Parch	0.025
alone	0.011

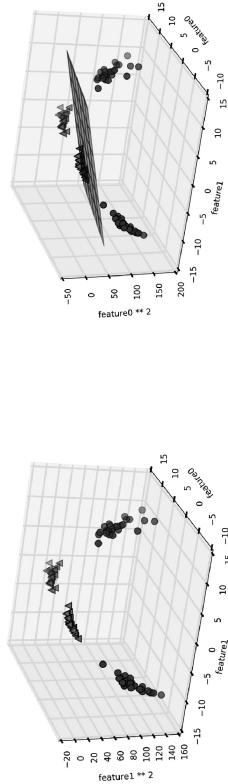


- Extensión del modelo de clasificador para conjuntos de datos más complejos con más dimensiones.
 - Los modelos lineales pueden enfrentar limitaciones en su flexibilidad, dadas las limitaciones de las líneas o hiper-planos.



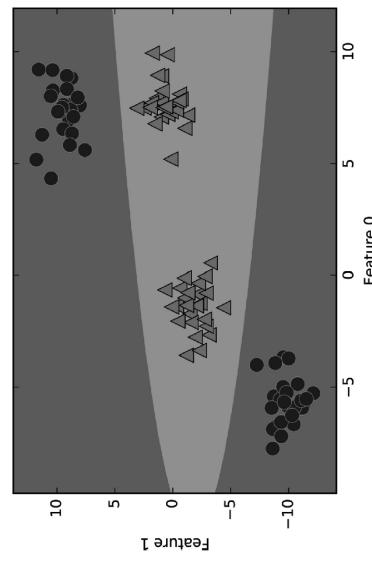
Kernel SVM

- Una forma de mejorar la flexibilidad es agregando más atributos, bajo criterios técnicos.
- Por ejemplo: agregar en set de datos anterior un tercer atributo llamado `(feature1 ** 2)`.
- Un plano en 3D es ahora capaz de separar adecuadamente las clases.



Kernel SVM

- Si llevamos el límite de decisión como una función de los 2 atributos originales, vemos como la linealidad se pierde y más bien se tiene una especie de ellipse.



Kernel SVM

- Agregar atributos no lineales puede hacer que los modelos lineales sean más efectivos.
- Hay que tener cuidado con agregar más atributos, en especial en términos del costo computacional.
- La clave: El uso de un **kernel**.
- Encontramos dos tipos de kernels muy comunes:
 - *Polynomial*: Se hace un cálculo de todos los posibles polinomios hasta un cierto grado de los atributos originales.
 - *RBF*: También conocido como kernel gaussiano, considera todos los posibles polinomios de todos los grados pero a mayor grado del polinomio menor relevancia tienen el atributo.

Kernel SVM

- Se dispone de varios hiper-parámetros de ajuste, donde los más comunes son:
 - *gamma*: Controla el ancho del kernel gaussiano, define la escala que determina la cercanía entre dos puntos.
 - *C*: Es el parámetro de regularización, limita el nivel de importancia de cada punto.

```
1 from sklearn.svm import SVC  
2 X, y = % dataset %  
3 svml = SVC(kernel='rbf', C=10, gamma=0.1).fit(X, y)
```


Kernel SVM

- Se desempeñan muy bien en la mayoría de conjuntos de datos.
 - Trabajan bien en conjuntos de pocas o muchas dimensiones.
 - Con muchas instancias o ejemplos de datos puede consumir muchos recursos computacionales.
 - El pre-procesamiento de los datos es muy común en kSVM, factor por el que random forest es muy usado.
 - Son más complejos de explicar y de inspeccionar con detalle durante la operación.

Questions?

