DigitalHouse > Coding School

DATA SCIENCE

MÓDULO 6

Feature Importance en árboles y ensambles

Evaluación de Modelos e Importancia de Features

OBJETIVOS DE LA CLASE

1. **Explicar cómo se calcula la importancia de los features para los árboles de decisión**
2. **Calcular importancia de los features manualmente**
3. **Extraer la importancia de los features con scikit-learn**
4. **Ampliar el cálculo a modelos de ensamble (RF, ET)**

Importancia de los Features en modelos No-Paramétricos

Introducción

- Cuando construimos un modelo de machine learning no sólo estamos interesados en la precisión de la predicción. Muy a menudo estamos construyendo un modelo para tener una idea sobre cuales son las variables predictoras relevantes.

- Por ejemplo, si tiene 1000 features para predecir la retención de usuarios.
  - ¿Qué features son relevantes?
  - ¿Cómo se pueden identificar?

Modelos Paramétricos

- Anteriormente se habló de la selección de features en el contexto de la Regresión Logística. La regresión logística es un modelo paramétrico, lo que significa que la hipótesis se describe en términos de coeficientes que ajustamos para mejorar la precisión del modelo.

- En particular, dado que la regresión logística es un modelo lineal, cada parámetro está asociado a un feature específico. Por lo tanto, si los features están normalizados, podemos interpretar el tamaño de cada coeficiente como indicador de la importancia relativa de ese feature específico, al menos sobre el log odds .

- También se vió como Scikit-Learn implementa varios métodos para la selección de features. Entre ellos podemos mencionar:
  - Eliminación recursiva de features
  - Selección de features basado en L1
  - Algoritmos genéticos

Árboles de decisión

- Los modelos basados en árboles no son paramétricos, por lo tanto no tenemos coeficientes para ajustar como lo hicimos en modelos lineales. Sin embargo aún nos puede interesar saber cuáles features son más importantes.

Árboles de decisión

- Los modelos basados en árboles no son paramétricos, por lo tanto no tenemos coeficientes para ajustar como lo hicimos en modelos lineales. Sin embargo aún nos puede interesar saber cuáles features son más importantes.

- Cuando se entrena un árbol, se puede calcular cuánto cada feature disminuye la impureza al **sumar todas las ganancias de pureza** donde ese feature es usado para determinar una división.

- En otras palabras, la importancia de un feature se calcula como la reducción total (normalizada) del criterio por ese feature. También se conoce como la importancia de Gini.

Random Forest

- ¿Cómo extendería la definición de la importancia de los features de los árboles de decisión a los Random Forest?

Random Forest

- ¿Cómo extendería la definición de la importancia de los features de los árboles de decisión a los Random Forest?
- Podemos calcular la importancia de los features de cada árbol en el bosque y luego promediar las importancias en todo el bosque.
- Un random forest es un conjunto de árboles entrenados en muestras aleatorias y subconjuntos aleatorios de features.
- Por lo tanto, para cada árbol la importancia de un feature se puede calcular usando el mismo procedimiento esbozado antes. Luego los resultados se promedian en todo el bosque.

Demo: Importancia de los Features en un Árbol de decisión

Demo: Importancia de los features en un Árbol de decisión

Tomemos nuevamente el dataset de autos y supongamos que estamos trabajando en una empresa de automóviles y que tenemos la tarea de identificar qué features impulsan la aceptación de un auto.

Hemos recogido algunos datos sobre varios features:

- PRECIO — precio total
  - buying — precio de compra
  - maint — precio del mantenimiento
- TÉCNICAS — features técnicos
  - CONFORT — confort
    - doors — número de puertas
    - persons — cantidad de personas que puede llevar
    - lug_boot — tamaño del baúl
  - safety — seguridad estimada del auto

Los valores de los features son:

buying	v-high, high, med, low
maint	v-high, high, med, low
doors	2, 3, 4, 5-more
persons	2, 4, more
lug_boot	small, med, big
safety	low, med, high

La distribución de clases (número de ocurrencias de cada clase)

clase	N	N[%]
unacc	1210	(70.023 %)
acc	384	(22.222 %)
good	69	(3.993 %)
v-good	65	(3.762 %)

Vamos a comenzar con la lectura del dataset de aceptabilidad de autos.

```
import pandas as pd
df = pd.read_csv('car.csv')
df.head()
```

	buying	maint	doors	persons	lug_boot	safety	acceptability
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

Esta vez codificaremos los features usando un esquema de codificación One Hot Encoding, es decir, los consideraremos como variables categóricas.
También necesitaremos asignarle números a las etiquetas del target. Para eso usaremos el LabelEncoder.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = LabelEncoder().fit_transform(df['acceptability'])
X = pd.get_dummies(df.drop('acceptability', axis=1))
X.ix[:,0:8].head()
```

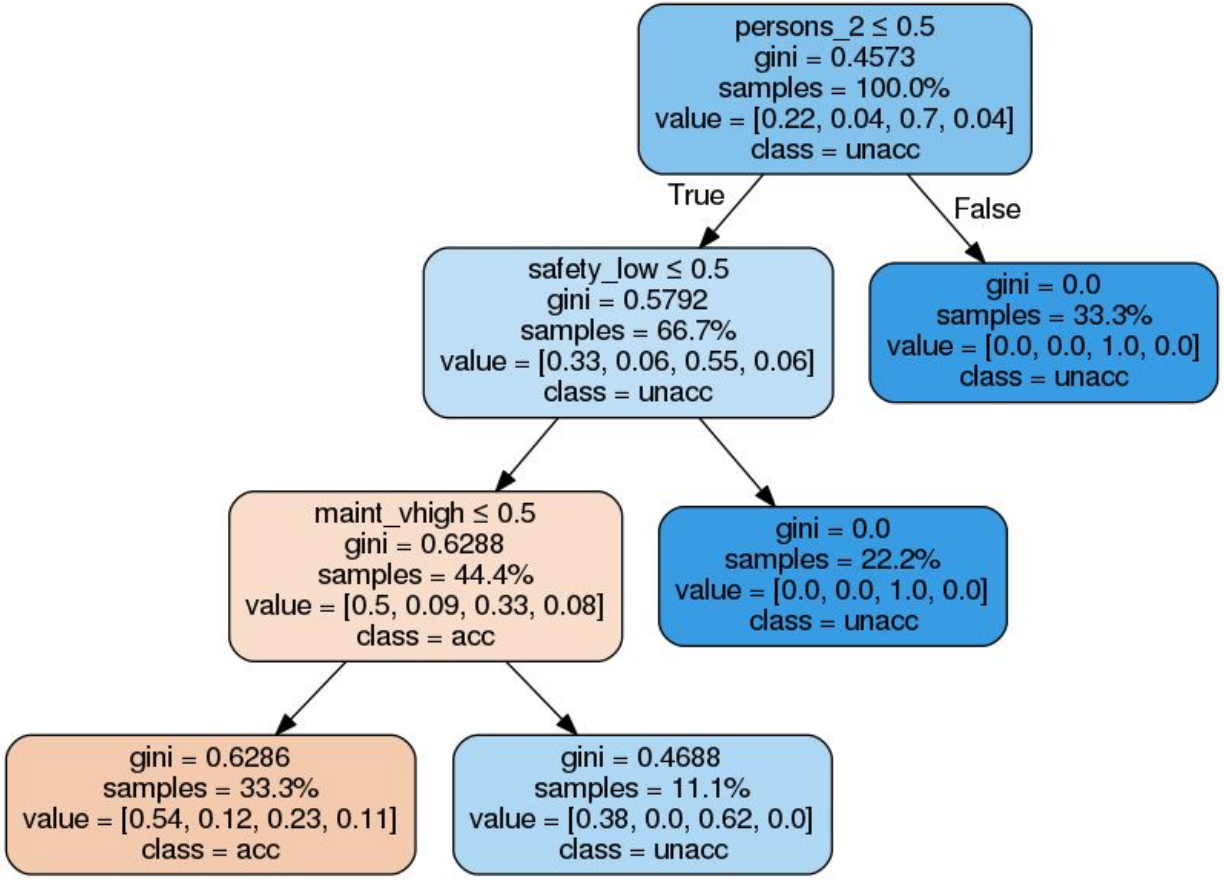
	buying_high	buying_low	buying_med	buying_vhigh	maint_high	maint_low	maint_med	maint_vhigh
0	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	0	1
2	0	0	0	1	0	0	0	1
3	0	0	0	1	0	0	0	1
4	0	0	0	1	0	0	0	1

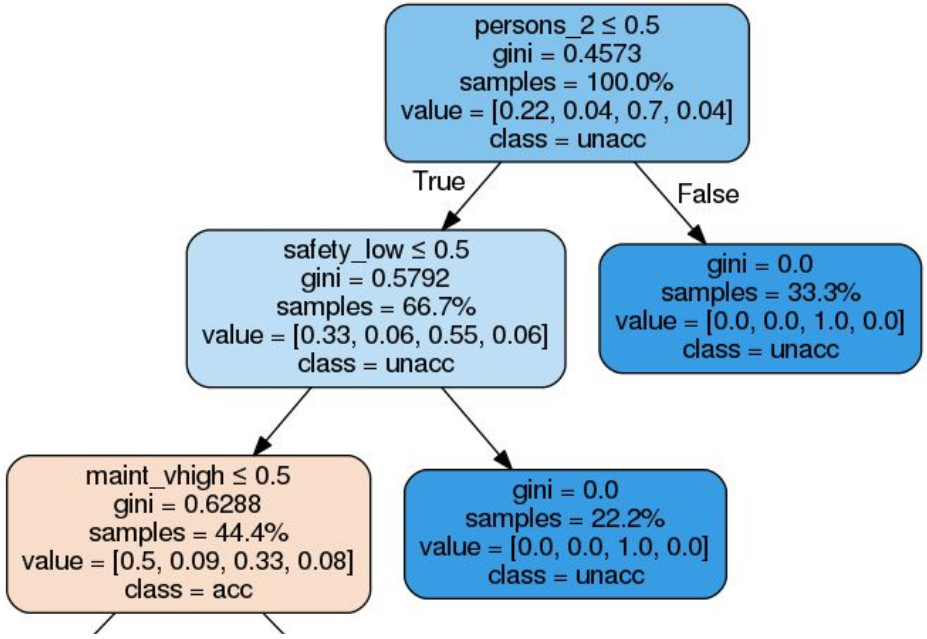
Vamos a entrenar un árbol de decisión en todo el conjunto de datos (ignoraremos el overfitting por el momento). También vamos a limitar artificialmente el árbol a ser pequeño para que podamos visualizarlo.

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(max_depth = 3, min_samples_split = 2, random_state = 11)
dt.fit(X, y)
```

Ahora visualicemos el árbol usando el exportador de graphviz.

```
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dot_data = export_graphviz(dt, out_file=None,
                           feature_names=X.columns, class_names=le.classes_,
                           filled=True, rounded=True, proportion=True, special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```


persons_2 ≤ 0.5
gini = 0.4573
samples = 100.0%
value = [0.22, 0.04, 0.7, 0.04]
class = unacc
True
False
safety_low ≤ 0.5
gini = 0.5792
samples = 66.7%
value = [0.33, 0.06, 0.55, 0.06]
class = unacc
gini = 0.0
samples = 33.3%
value = [0.0, 0.0, 1.0, 0.0]
class = unacc
maint_vhigh ≤ 0.5
gini = 0.6288
samples = 44.4%
value = [0.5, 0.09, 0.33, 0.08]
class = acc
gini = 0.0
samples = 22.2%
value = [0.0, 0.0, 1.0, 0.0]
class = unacc
gini = 0.6286
samples = 33.3%
value = [0.54, 0.12, 0.23, 0.11]
class = acc
gini = 0.4688
samples = 11.1%
value = [0.38, 0.0, 0.62, 0.0]
class = unacc



La primer opción involucra person_2. Si el auto sólo acepta 2 personas (person_2 == 1) entonces la clase es inaceptable. Esto ocurre en el 33% de los casos. Observar que la hoja bajo la rama Falso es 100% pura y por lo tanto su medida Gini es 0.0.

Por otro lado, si el auto puede tener más de 2 personas, tendremos que considerar otras opciones. Por ejemplo, si el auto no es seguro, entonces también es inaceptable. Y así sucesivamente.

Si el objetivo es una clasificación con K clases posibles.

Si el nodo *m* representa una región *Rm* con *Nm* muestras, donde *Nk* son muestras de la clase *k*, la proporción de la clase *k* en nodo *m* puede escribirse como:

$$C_k = \frac{N_k}{N_m}$$

El índice de impureza Gini se define entonces como:

$$\text{Gini} = \sum_{k=0}^{K-1} C_k(1 - C_k) = 1 - \sum_{k=0}^{K-1} C_k^2$$

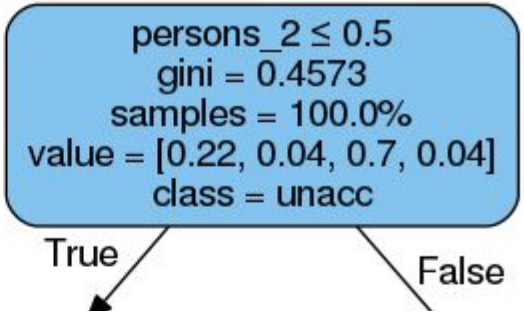
Grafiquemos el índice Gini para varias proporciones en una clasificación binaria

```
import numpy as np
import matplotlib.pyplot as plt
C0 = np.linspace(0,1)
C1 = 1.0 - C0
gini = 1 - ( C0**2 + C1**2 )
plt.plot(C0, gini)
```

Aquí vamos a verificar el cálculo del índice Gini en el nodo raíz del árbol.

```
cantidad_ocurrencias_clases = pd.Series(y).value_counts()
total_observaciones = sum( cantidad_ocurrencias_clases )
proporciones_clases = cantidad_ocurrencias_clases / total_observaciones
print "Proporciones Clases"
print proporciones_clases

gini = 1 - sum( proporciones_clases ** 2 )
print "Gini: ",gini
```

```
Proporciones Clases
2    0.700231
0    0.222222
1    0.039931
3    0.037616
dtype: float64
Gini:  0.457283763074
```



Ahora veamos la importancia de los features del árbol generado.

Para lo cual utilizaremos `feature_importances_`

```
importancia_features = pd.DataFrame( dt.feature_importances_ , index = X.columns,
    columns=['importancia']
  ).sort_values( 'importancia',ascending=False)
importancia_features.head()
```

	importancia
safety_low	0.545239
persons_2	0.363493
maint_vhigh	0.091268
buying_high	0.000000
doors_5more	0.000000

Observar que solo tres features tienen alguna importancia. Esto se debe a que solamente esos tres fueron utilizados para hacer divisiones, dado que restringimos artificialmente el árbol ser pequeño.

Ahora verifiquemos el cálculo de la importancia.

```
ganancia_gini_persons_2    = 1.000 * 0.4573 - 0.667 * 0.5792 - 0.333 * 0.0000
ganancia_gini_safety_low   = 0.666 * 0.5792 - 0.444 * 0.6288 - 0.222 * 0.0000
ganancia_gini_buying_vhigh = 0.444 * 0.6288 - 0.333 * 0.6286 - 0.111 * 0.4688

norm = ganancia_gini_persons_2 + ganancia_gini_safety_low + ganancia_gini_buying_vhigh

print "persons_2:", ganancia_gini_persons_2 / norm
print "safety_low:", ganancia_gini_safety_low / norm
print "buying_vhigh:", ganancia_gini_buying_vhigh / norm
```

```
persons_2: 0.363296106372
safety_low: 0.545453987045
buying_vhigh: 0.0912499065828
```

Práctica guiada: Importancia de los features en modelos Ensamble

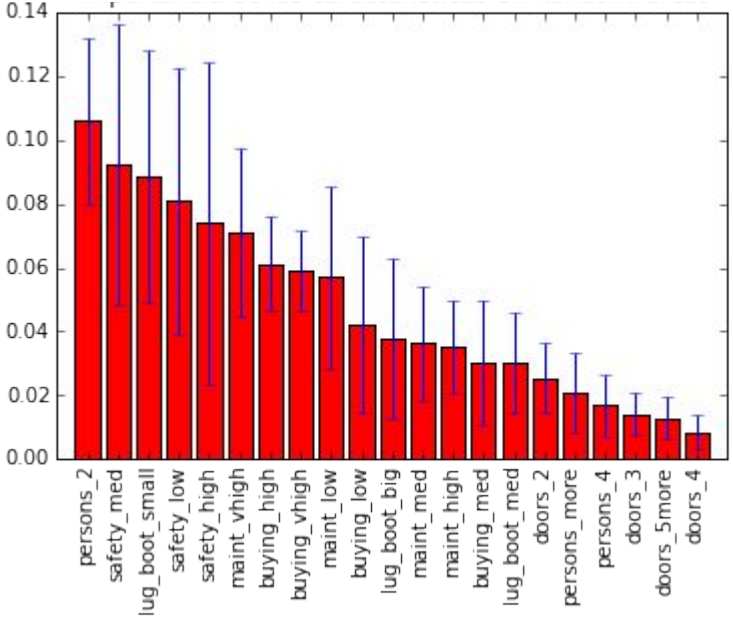
Scikit Learn implementa importancia de features en random forest y extra trees.
Vamos a entrenar ambos modelos e investigar la importancia de los features.

```
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
rf = RandomForestClassifier(class_weight='balanced', n_jobs=-1)
rf.fit(X, y)
```

El Random forest expone la importancia de los features y lo calcula como la media de la importancia de los features de los árboles base. Vamos a verificar eso.

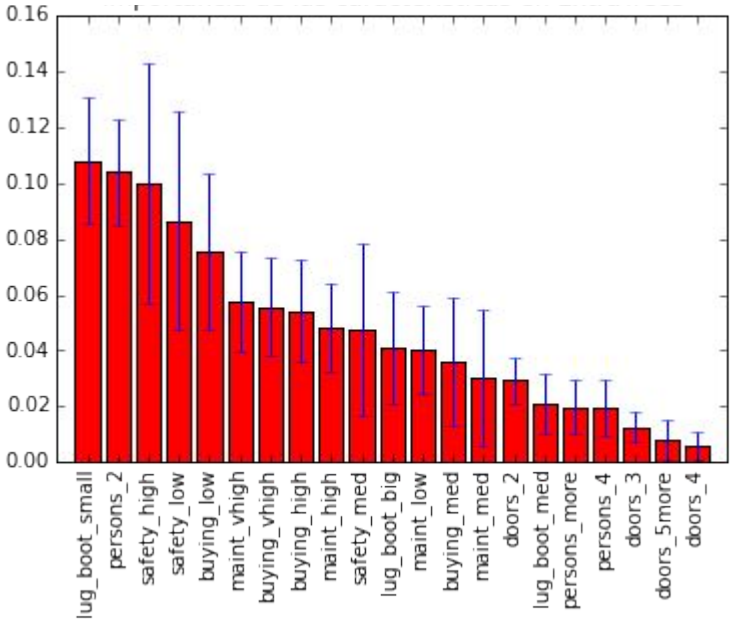
```
importancias = rf.feature_importances_
print all(importancias == np.mean([tree.feature_importances_ for tree in rf.estimators_], axis=0))
True
```

```
std = np.std([tree.feature_importances_ for tree in rf.estimators_], axis=0)
indices = np.argsort(importancias)[::-1]
nombres_features = X.columns

# Graficar la importancia de los features
 en el random forest
plt.figure()
plt.title("Importancia de los features")
plt.bar(range(X.shape[1]), importancias[indices],
      color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]),
      nombres_features[indices], rotation=90)
plt.xlim([-1, X.shape[1]])
plt.show()
```



Aquí calculamos la importancia de los features para ExtraTrees

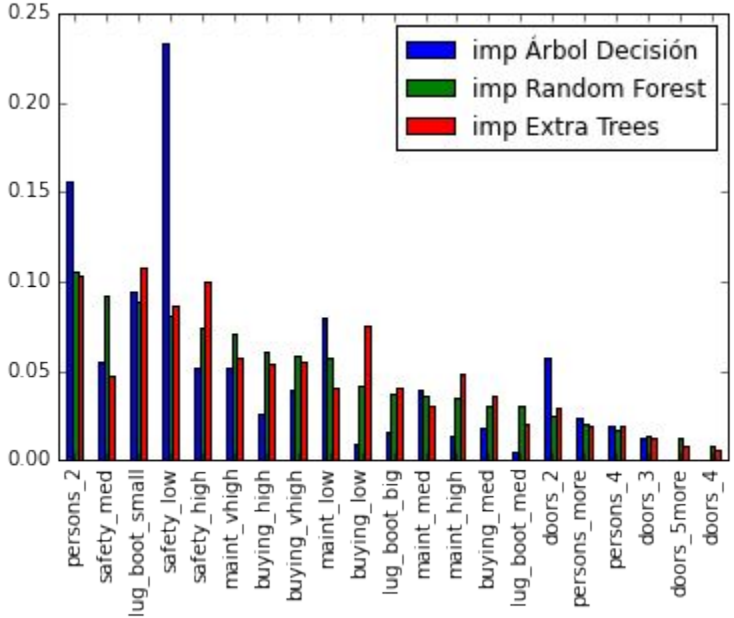
```
et = ExtraTreesClassifier(class_weight='balanced', n_jobs=-1)
et.fit(X, y)
importancias = et.feature_importances_
std = np.std([tree.feature_importances_
              for tree in et.estimators_], axis=0)

indices = np.argsort(importancias)[::-1]
nombres_features = X.columns

# Graficar la importancia de los features
plt.figure()
plt.title("Importancia de los features en ExtraTrees")
plt.bar(range(X.shape[1]), importancias[indices],
       color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), nombres_features[indices],
       rotation=90)
plt.xlim([-1, X.shape[1]])
plt.show()
```



Finalmente comparemos los 3 modelos

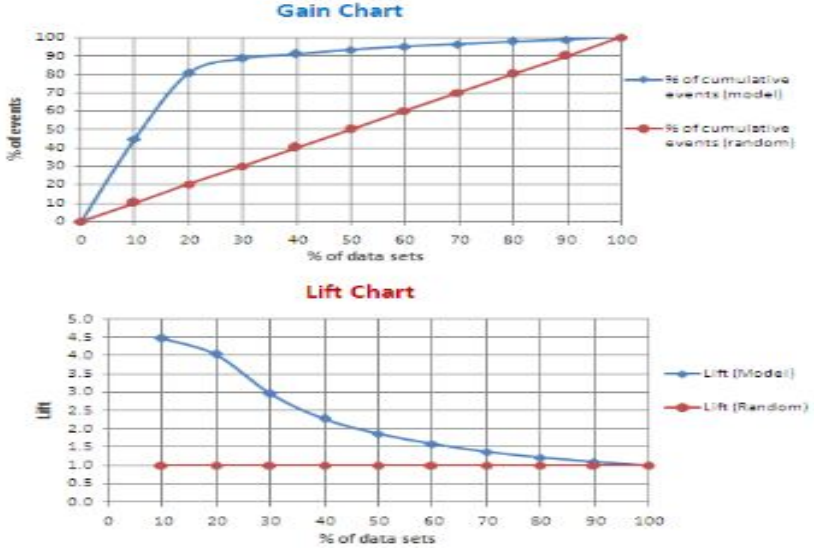
```
dt2 = DecisionTreeClassifier()
dt2.fit(X, y)
importancias = pd.DataFrame(zip(dt2.feature_importances_,
        rf.feature_importances_,et.feature_importances_),
    index=X.columns, columns=['imp Árbol Decisión',
    'imp Random Forest', 'imp Extra Trees']).sort_values(
        'imp Random Forest', ascending=False)
importancias.plot(kind='bar')
importancias.head()
```

	imp Árbol Decisión	imp Random Forest	imp Extra Trees
persons_2	0.155542	0.106006	0.103813
safety_med	0.055551	0.092098	0.047409
lug_boot_small	0.094315	0.088645	0.107911
safety_low	0.233312	0.081021	0.086503
safety_high	0.051794	0.073872	0.099826

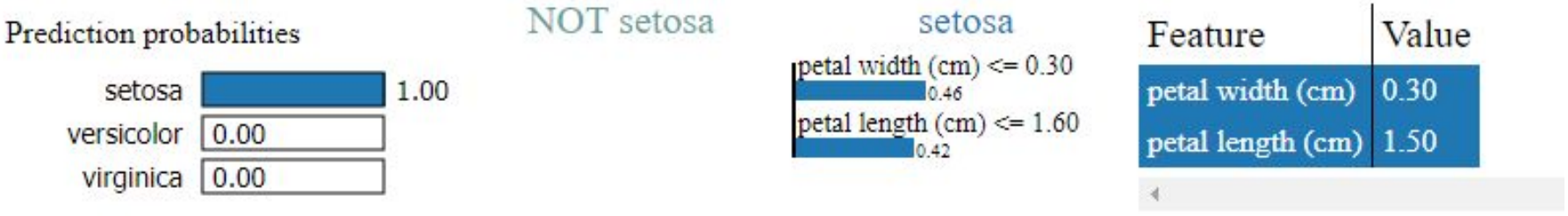


¿Qué feature es relevante para la predicción de una observación individual?

- Hasta aquí aprendimos sobre la importancia de los features para el total de la muestra.
- Con el gain chart podríamos identificar cuáles observaciones podemos “intervenir” para detectar por ejemplo la baja de un servicio.



- ¿Pero qué podemos decir de una observación individual?
- Intuición: aproximar linealmente la función de decisión del modelo en la vecindad de una instancia (punto). LIME



Práctica guiada: LIME

CONCLUSIÓN

- Hasta aquí aprendimos sobre la importancia de los features y cómo se calculan para los modelos basados en árboles.

- También hemos profundizado nuestra comprensión de la medidas Gini.