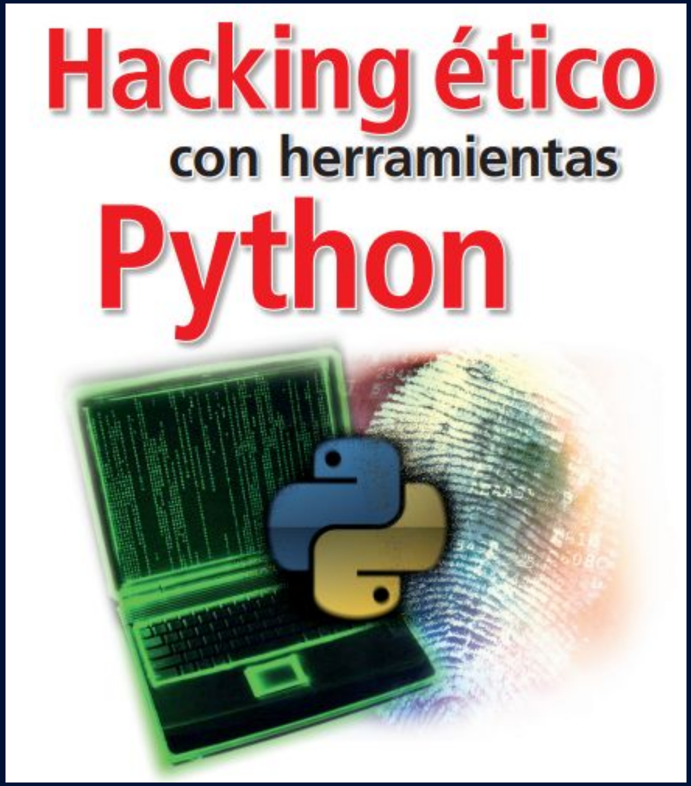
UA | October 4 - 6, 2019

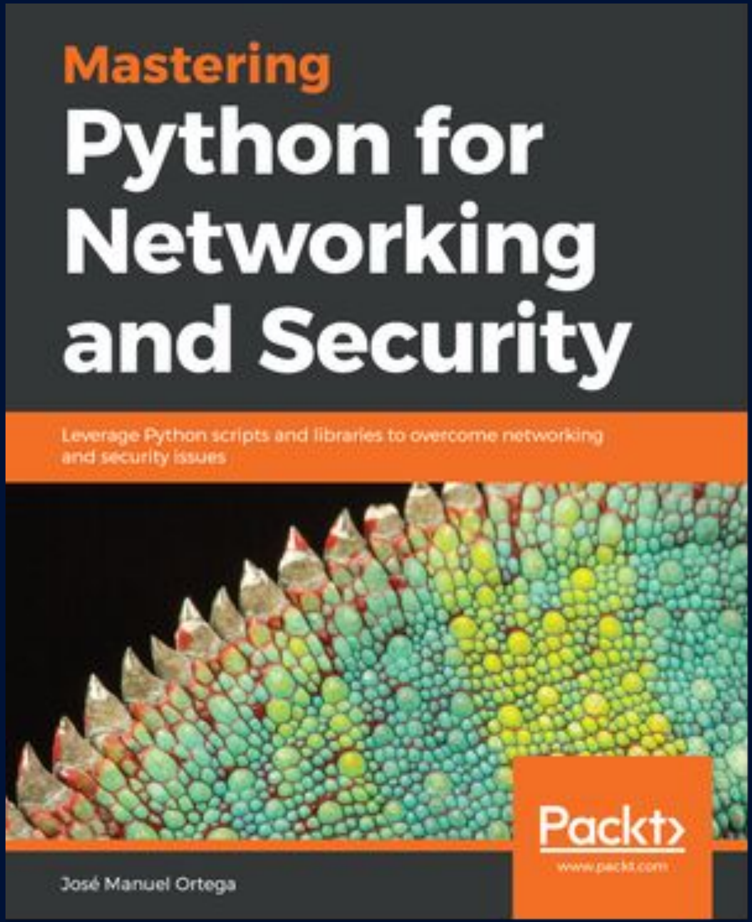
Machine learning para proyectos de seguridad

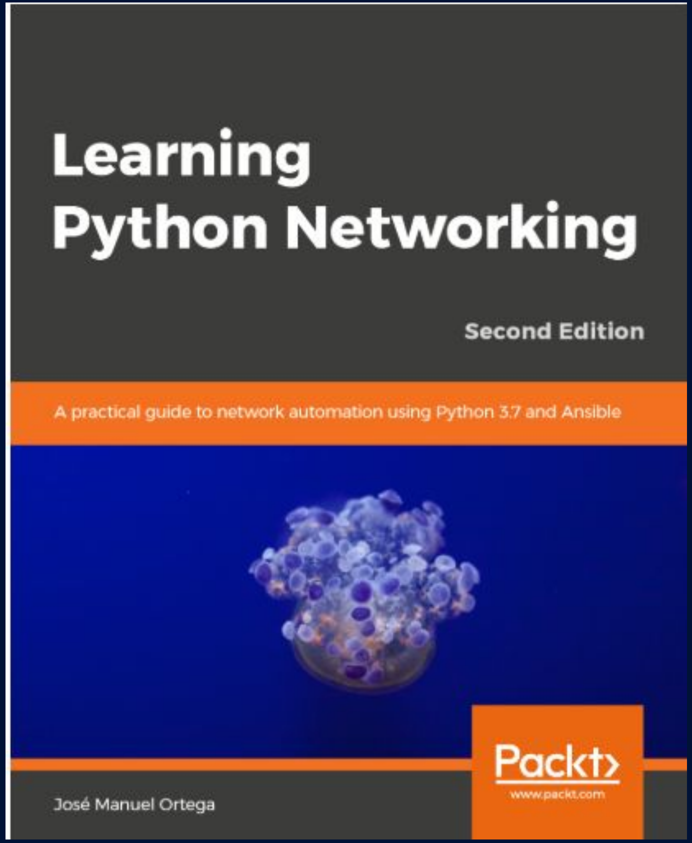
José Manuel Ortega @jmortegac

ALICANTE 2019

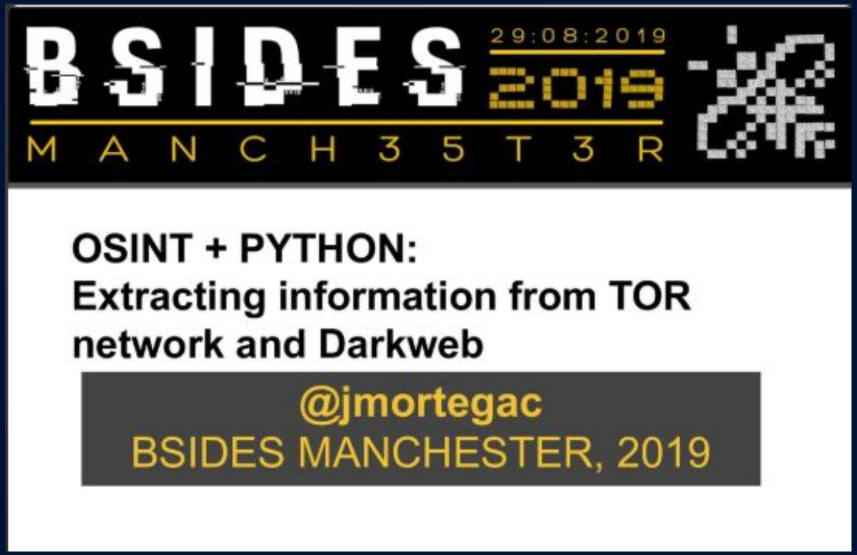
About me


Hacking ético
con herramientas
Python


Mastering
Python for Networking and Security
Leverage Python scripts and libraries to overcome networking and security issues
Packt>
www.packt.com
José Manuel Ortega


Learning
Python Networking
Second Edition
A practical guide to network automation using Python 3.7 and Ansible
Packt>
www.packt.com
José Manuel Ortega

About me



SCAN ME

Agenda

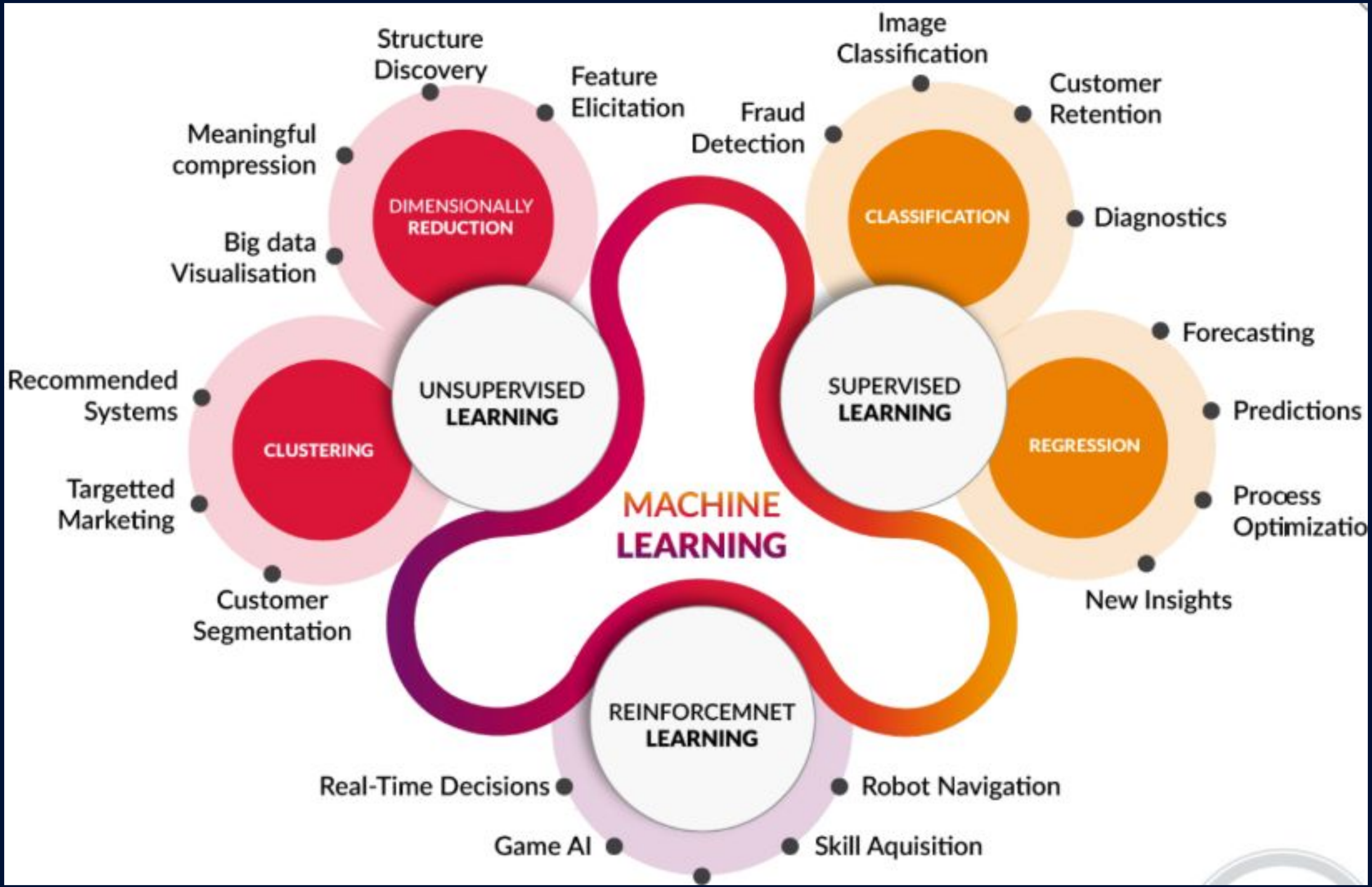
- Introducción al Machine Learning
- Algoritmos y SKLearn con python
- Casos de uso(Spam,fraude)
- Detección de anomalías
- Conclusiones y recursos

AI vs ML

La **inteligencia artificial** es un término utilizado para describir un sistema que percibe su entorno y toma medidas para maximizar las posibilidades de lograr sus objetivos.

El **aprendizaje automático** es un conjunto de técnicas que permiten a las computadoras realizar tareas sin ser programadas explícitamente. Los sistemas de ML generalizan a partir de datos pasados para hacer predicciones sobre datos futuros.

Tipos de ML



MACHINE LEARNING
UNSUPERVISED LEARNING
SUPERVISED LEARNING
REINFORCEMNET LEARNING
DIMENSIONALLY REDUCTION
CLUSTERING
CLASSIFICATION
REGRESSION
Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
Recommended Systems
Targetted Marketing
Customer Segmentation
Image Classification
Customer Retention
Fraud Detection
Diagnostics
Forecasting
Predictions
Process Optimizatio
New Insights
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition

Tipos de ML

El **aprendizaje supervisado** se centra en modelos que predicen las probabilidades de nuevos eventos en función de las probabilidades de eventos observados previamente. Por ejemplo: **determinar si un archivo es malware o no.**

Los modelos de **aprendizaje no supervisado** intentan encontrar patrones en datos no etiquetados. Por ejemplo : **determinar cuántas familias de malware existen en el conjunto de datos y qué archivos pertenecen a cada familia.**

Aprendizaje supervisado

Clasificación: Los algoritmos de clasificación predicen a qué categoría pertenece una entrada en función de las probabilidades aprendidas de las entradas observadas previamente. **Por ejemplo: determinar si un archivo es malware o no.**

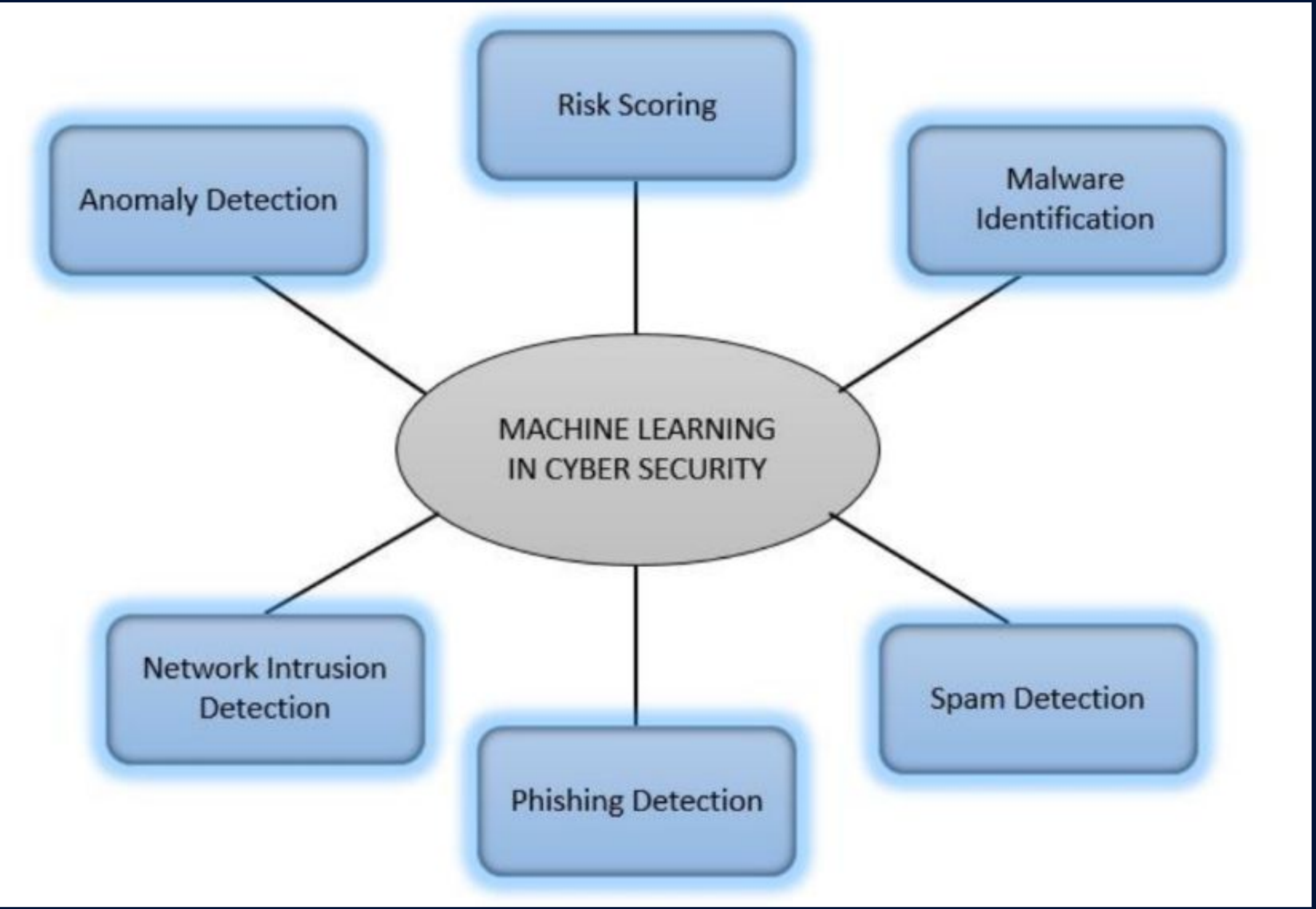
Regresión: los modelos de regresión (lineal, logística) predicen un valor de salida continuo para una entrada determinada en función de los valores de salida asociados con las entradas anteriores. **Por ejemplo: predecir cuántas muestras de malware se detectarán el próximo mes.**

Aprendizaje no supervisado

Clustering:Consiste en agrupar un conjunto de objetos de tal manera que los objetos en el mismo grupo(cluster) sean más similares entre sí que con los de otros grupos

Detección de anomalías

ML en seguridad



Proceso de ML

Construir un modelo

- **Recopilar** muestras de datos de ambas clasificaciones para entrenar el modelo de aprendizaje automático.
- **Extraer** características de cada ejemplo de entrenamiento para representar el ejemplo numéricamente.
- **Entrenar** al sistema de aprendizaje automático para identificar elementos que sigan un patrón específico.
- **Probar** el sistema con datos que no se utilizaron durante el entrenamiento para evaluar su precisión o accuracy.

Extracción características

E-mail

Hi James

Do you need to go to Walmart ? Take that gift card below now:

Download it now

Walmart Team

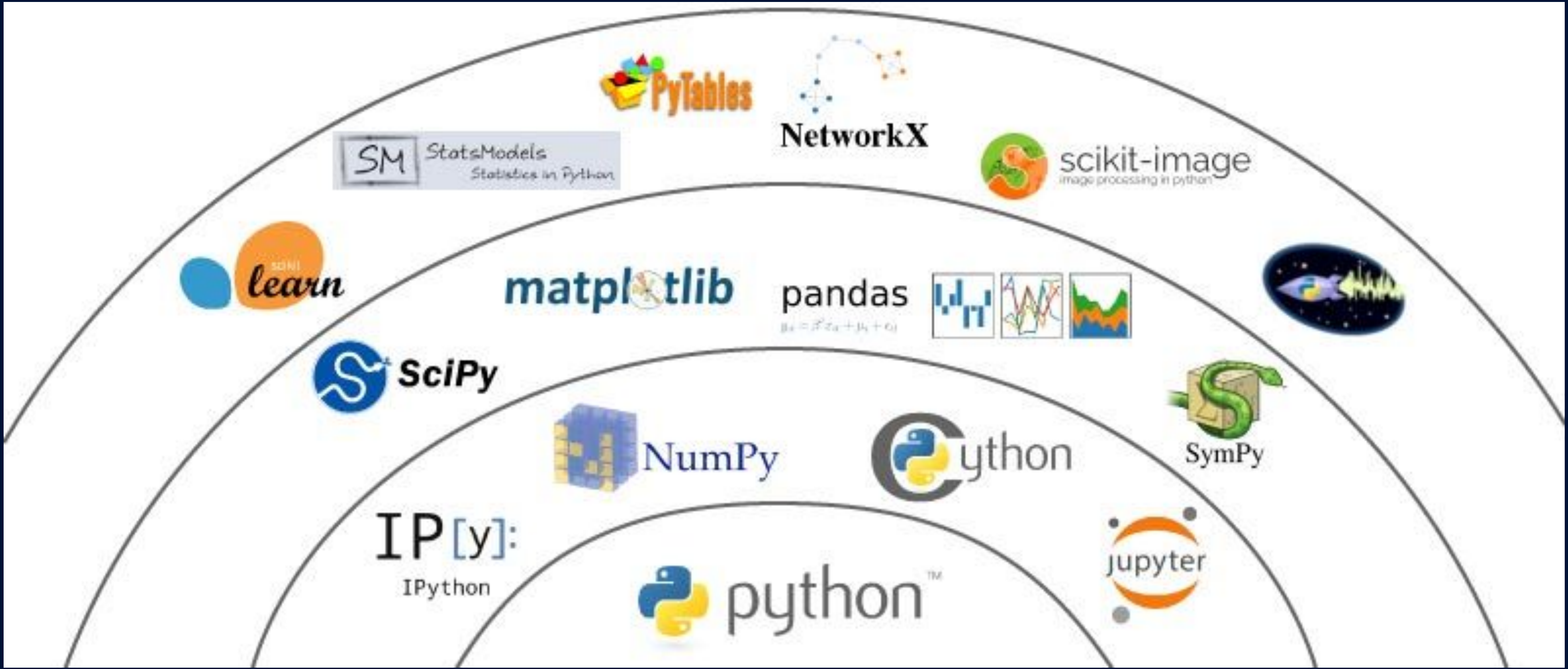
Feature	Count
do	1
gift	1
go	1
card	1
James	1
Walmart	2
?	1

URL

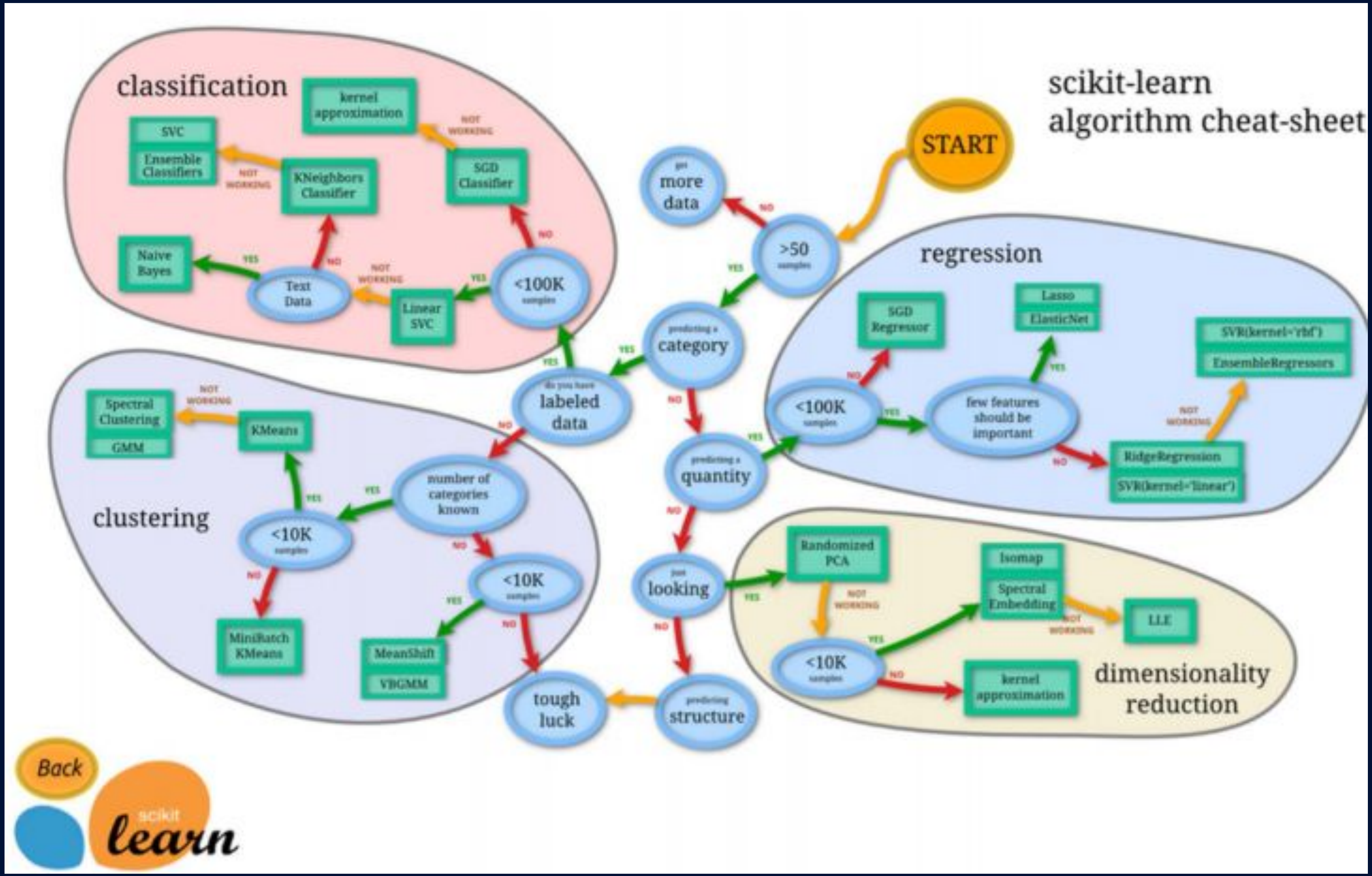
http://admin.jablum.cz/files/2914d7d2a19d2EyYWE=/customer_center/customer-IDPP00C741/myaccount/signin/

Feature	Count
URL Length	102
/ count	6
= count	1
? count	0
& count	0

python Machine learning



Sklearn



scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
kernel approximation
SVC
Ensemble Classifiers
KNeighbors Classifier
SGD Classifier
Naive Bayes
Text Data
Linear SVC
<100K samples
clustering
Spectral Clustering
GMM
KMeans
number of categories known
<10K samples
MiniBatch KMeans
MeanShift
VBGMM
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
few features should be important
RidgeRegression
SVR(kernel='linear')
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
YES
NO
NOT WORKING
Back
scikit
learn

Selección de características

scikit learn

Home | Installation | Documentation | Examples

Google Custom Search

Previous sklearn.decom... | **Next** sklearn.decom... | **Up** API Reference

scikit-learn v0.21.3
Other versions

Please **cite us** if you use the software.

`sklearn.decomposition` .PCA
Examples using `sklearn.decomposition.PCA`

`sklearn.decomposition`.PCA

class `sklearn.decomposition.` **`PCA`** (*n_components=None*, *copy=True*, *whiten=False*, *svd_solver='auto'*, *tol=0.0*, *iterated_power='auto'*, *random_state=None*) [source]

Principal component analysis (PCA)

Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space. The input data is centered but not scaled for each feature before applying the SVD.

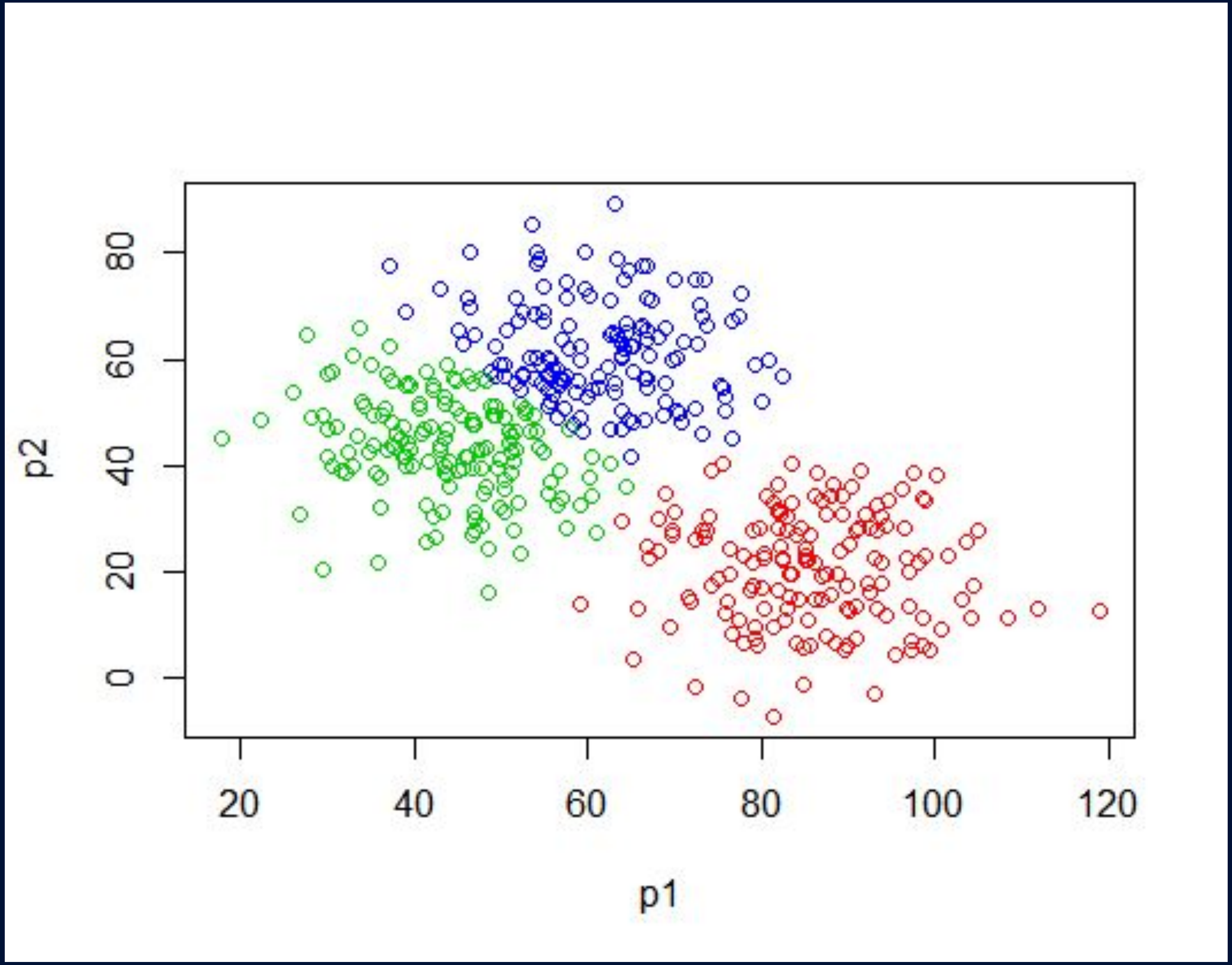
It uses the LAPACK implementation of the full SVD or a randomized truncated SVD by the method of Halko et al. 2009, depending on the shape of the input data and the number of components to extract.

It can also use the scipy.sparse.linalg ARPACK implementation of the truncated SVD.

Notice that this class does not support sparse input. See `TruncatedSVD` for an alternative with sparse data.

Read more in the User Guide.

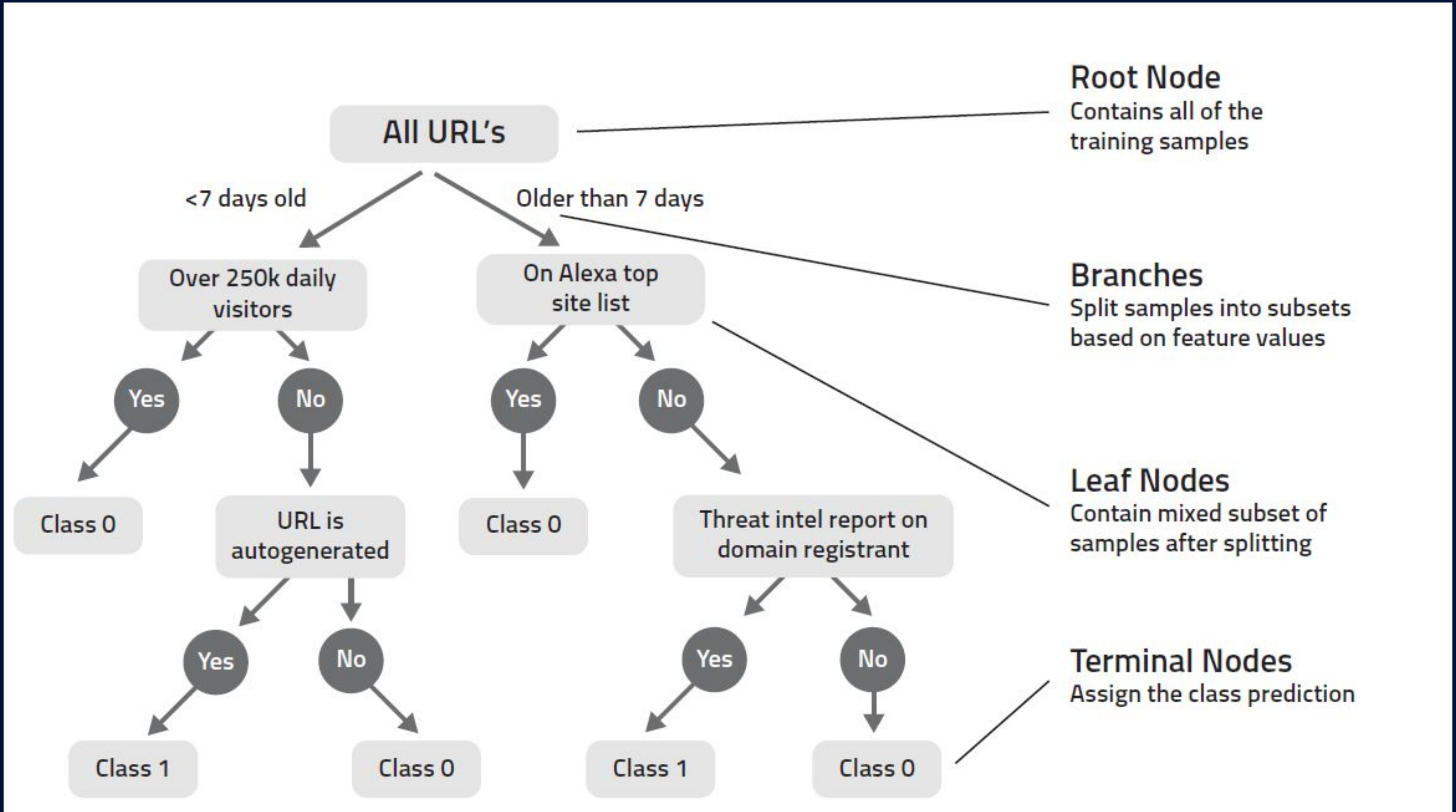
Clustering



Clustering Sklearn

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large n_samples, medium n_clusters withMiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium n_samples, small n_clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large n_samples and n_clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large n_samples, medium n_clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large n_clustersand n_samples	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Árboles de decisión



Sklearn

- El proceso consiste en:
  - Elegir el modelo.
  - Seleccionar los hiperparámetros.
  - Extraer la matriz de características y vector de predicción.
  - Ajustar el modelo a los datos (entrenamiento).
  - Predecir etiquetas para datos desconocidos.

Módulos python

- import numpy as np
- import pandas as pd
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import accuracy_score , confusion_matrix

Entrenar nuestro modelo

- **In [19]: from sklearn.tree import DecisionTreeClassifier**
- **In [21]: model = DecisionTreeClassifier()**
- **In [22]: X_train , X_test , y_train , y_test = train_test_split (X , y , test_size =0.2, random_state =1)**
- **In [23]: model.fit (X_train ,y_train);**

Evaluar nuestro modelo

- In [23]: y_pred = model.predict (X_test)
- In [24]: accuracy_score (y_pred , y_test)
- Out [24]: 0.9745454545454545

Evaluar nuestro modelo

- **from sklearn.metrics import confusion_matrix**
- **print(confusion_matrix(Y_test,Y_pred))**

Matriz de confusión

		Valor real	
		Fraude	Legítimo
Valor predicción	Fraude	Verdaderos Positivos	Falsos Positivos
	Legítimo	Falsos Negativos	Verdaderos Negativos

Métricas

Precisión (*precision*): Se calcula dividiendo el número de verdaderos positivos por la suma del número de verdaderos positivos y el número de falsos positivos

$$Precision = \frac{VeraderosPositivos}{VerdaderosPositivos + FalsosPositivos}$$

Exhaustividad (*recall*): Se calcula dividiendo el número de verdaderos positivos por la suma del numero de verdaderos positivos y el número de falsos negativos

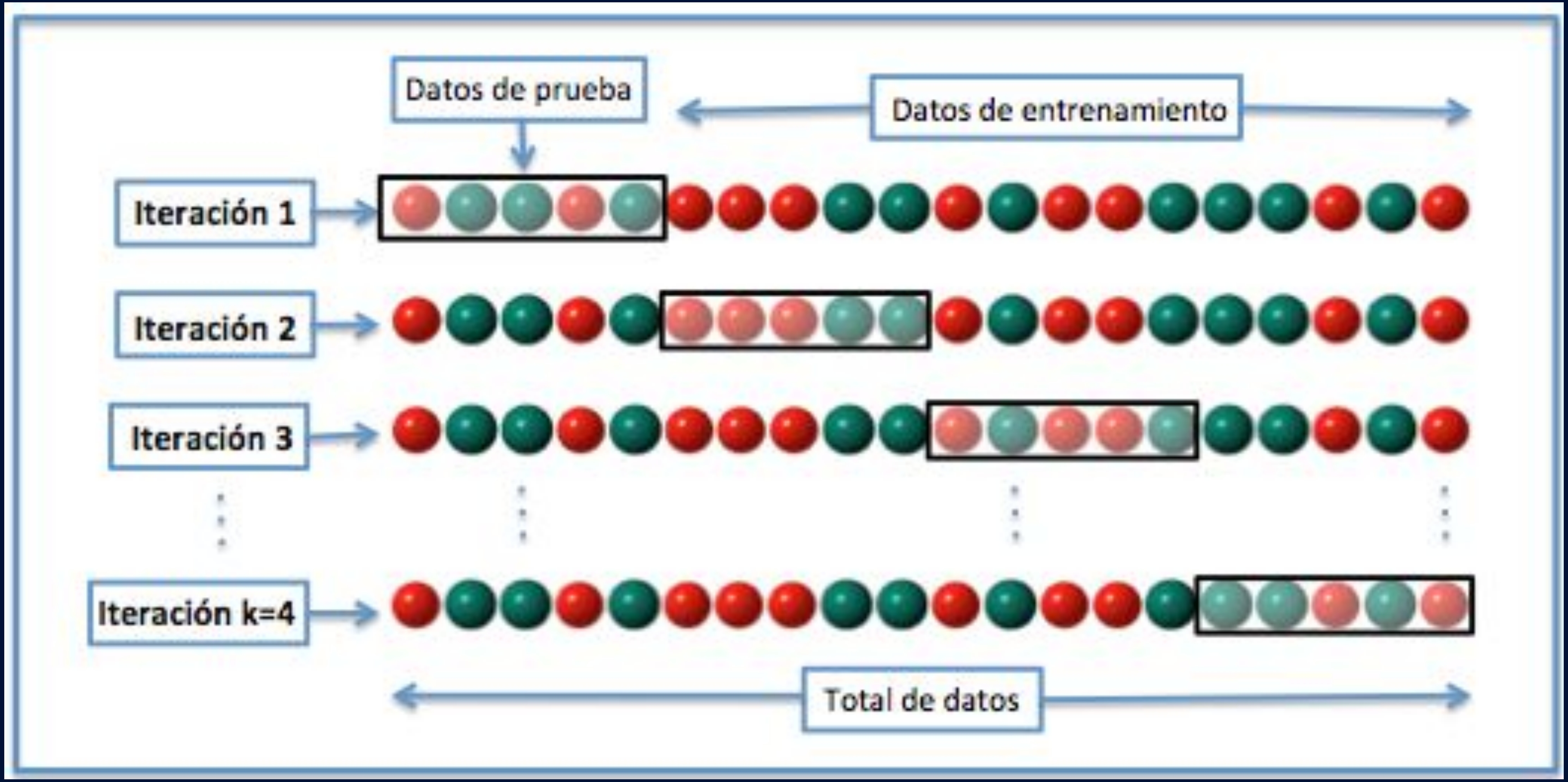
$$Recall = \frac{VeraderosPositivos}{VerdaderosPositivos + FalsosNegativos}$$

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

Sobreentrenamiento

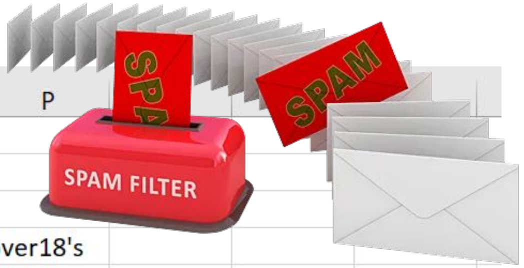
- **Para evitar el sobreajuste se divide el dataset en dos partes:**
    - **Datos de entrenamiento**
    - **Datos de evaluación**
- **K-fold cross validation**

Cross-validation



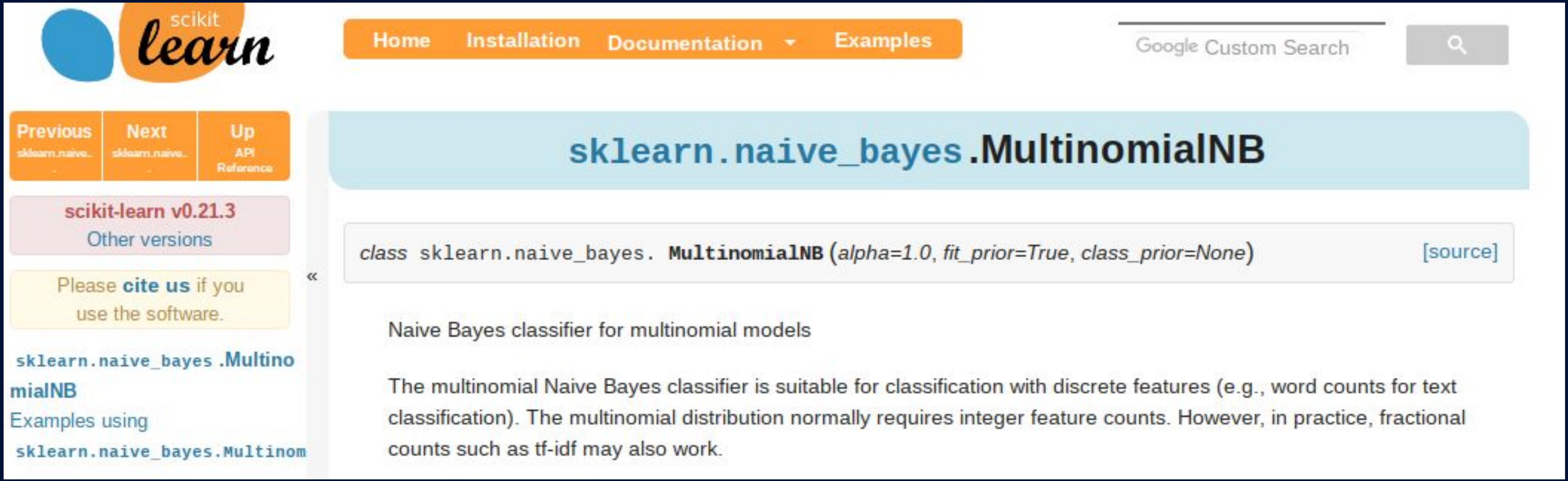
Detección de spam

https://www.kaggle.com/ishansoni/sms-spam-collection-dataset



	A	B
1	v1	v2
2	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
3	ham	Ok lar... Joking wif u oni...
4	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
5	ham	U dun say so early hor... U c already then say...
6	ham	Nah I don't think he goes to usf, he lives around here though
7	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, å£1.50 to rcv
8	ham	Even my brother is not like to speak with me. They treat me like aids patent.
9	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
10	spam	WINNER!! As a valued network customer you have been selected to receivea å£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
11	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
12	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
13	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
14	spam	URGENT! You have won a 1 week FREE membership in our å£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
15	ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times
16	ham	I HAVE A DATE ON SUNDAY WITH WILL!!
17	spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
18	ham	Oh k...i'm watching here:)
19	ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
20	ham	Fine if thatåÕs the way u feel. ThatåÕs the way its gota b

Detección de spam



Home Installation Documentation Examples

Google Custom Search

Previous Next Up API Reference

scikit-learn v0.21.3
Other versions

Please **cite us** if you use the software.

`sklearn.naive_bayes`.MultinomialNB
Examples using `sklearn.naive_bayes.Multinom`

`sklearn.naive_bayes`.MultinomialNB

class `sklearn.naive_bayes.` **`MultinomialNB`** (*alpha=1.0*, *fit_prior=True*, *class_prior=None*) [source]

Naive Bayes classifier for multinomial models

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

Detección de spam

```
In [7]: from sklearn.feature_extraction.text import CountVectorizer

        vectorizer = CountVectorizer()
        X_train_vector = vectorizer.fit_transform(X_train)
        X_test_vector = vectorizer.transform(X_test)
```

```
In [8]: from sklearn.naive_bayes import MultinomialNB
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import classification_report

        # Initialize the classifier and make label predictions
        mnb = MultinomialNB()
        mnb.fit(X_train_vector, y_train)
        y_pred = mnb.predict(X_test_vector)

        # Print results
        print(classification_report(y_test, y_pred, target_names=['Spam', 'Ham']))
        print('Classification accuracy {:.1%}'.format(accuracy_score(y_test, y_pred)))
```

```
             precision    recall  f1-score   support

       Spam       0.99      0.94      0.97     15035
        Ham       0.90      0.98      0.94      7591

avg / total       0.96      0.96      0.96     22626

Classification accuracy 95.6%
```

Detección de spam

```python
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model.logistic import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score

dataframe = pd.read_csv('SMSSpamCollectionDataSet', delimiter='\t',header=None)

X_train_dataset, X_test_dataset, y_train_dataset, y_test_dataset = train_test_split(dataframe[1],dataframe[0])

vectorizer = TfidfVectorizer()
X_train_dataset = vectorizer.fit_transform(X_train_dataset)
classifier_log = LogisticRegression()
classifier_log.fit(X_train_dataset, y_train_dataset)

X_test_dataset = vectorizer.transform( ['URGENT! Your Mobile No 1234 was awarded a Prize', 'Hey honey, whats up?'] )

predictions_logistic = classifier.predict(X_test_dataset)
print(predictions)
```

Detección de spam

```python
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3,
                                                    random_state = 0)

print(X_train)
# Create bag of words
X_train = countvec.fit_transform(X_train)
X_test = countvec.transform(X_test)

# Number of features before PCA
print('Features before PCA: {}'.format(X_train.shape[1]))
#Features before PCA: 1000
# Train PCA model
pca = PCA(n_components=200)
X_train_reduced = pca.fit_transform(X_train.toarray())
print(X_train_reduced)
# Training happens only on train data
# Transforming test data with pca model trained from train data
X_test_reduced = pca.transform(X_test.toarray())

# Number of features after PCA
print('Features after PCA: {}'.format(X_train_reduced.shape[1]))
#Features after PCA: 200
# Build Model
lr = LogisticRegression(penalty='l2')
lr.fit(X_train_reduced, y_train)
y_pred = lr.predict(X_test_reduced)
score = accuracy_score(y_test, y_pred)
print("Accuracy score: {}%".format(round(score*100)))
print(classification_report(y_test,y_pred,target_names=['Spam','Ham']))
```

```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

Accuracy score: 97.0%

	precision	recall	f1-score	support
Spam	0.98	1.00	0.99	1434
Ham	0.97	0.85	0.91	238
accuracy			0.97	1672
macro avg	0.97	0.92	0.95	1672
weighted avg	0.97	0.97	0.97	1672

Detección de fraude

accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	label
29	1	4.745402	paypal	28.2048611111	0
26	1	4.745402	paypal	0.0	0
3	1	5.034622	creditcard	0.0	0
1	1	4.748314	creditcard	0.00277777777778	1

Detección de fraude

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,accuracy_score

df = pd.read_csv('payment_fraud.csv')
print(df.sample(3))

df = pd.get_dummies(df,columns=['paymentMethod'])

Y = df['label']
X = df.drop('label',axis=1)

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3)

clf = LogisticRegression()
clf.fit(X_train,Y_train)
Y_pred = clf.predict(X_test)

print(confusion_matrix(Y_test,Y_pred))
print("Precision",accuracy_score(Y_test,Y_test))
```

		Valor real	
		Legítimo	Fraude
Valor predicción	Legítimo	12753	0
	Fraude	1	189

Detección de fraude

scikit learn

Home | Installation | Documentation | Examples

Google Custom Search

Previous 1.12. Multicl... | Next 1.14. Semi-Supervised | Up 1. Supervised...

scikit-learn v0.21.3
Other versions

Please **cite us** if you use the software.

- 1.13. Feature selection
- 1.13.1. Removing features with low variance
- 1.13.2. Univariate feature selection
- 1.13.3. Recursive feature elimination
- 1.13.4. Feature selection using SelectFromModel
  - 1.13.4.1. L1-based feature selection
  - 1.13.4.2. Tree-based feature

1.13. Feature selection

The classes in the `sklearn.feature_selection` module can be used for feature s
sample sets, either to improve estimators' accuracy scores or to boost their performar

1.13.1. Removing features with low variance

`VarianceThreshold` is a simple baseline approach to feature selection. It removes a
meet some threshold. By default, it removes all zero-variance features, i.e. features th

As an example, suppose that we have a dataset with boolean features, and we want t
one or zero (on or off) in more than 80% of the samples. Boolean features are Bernou
of such variables is given by

$$\mathrm{Var}[X] = p(1-p)$$

so we can select using the threshold `.8 * (1 - .8)`:

sklearn.feature_selection: Feature Selection

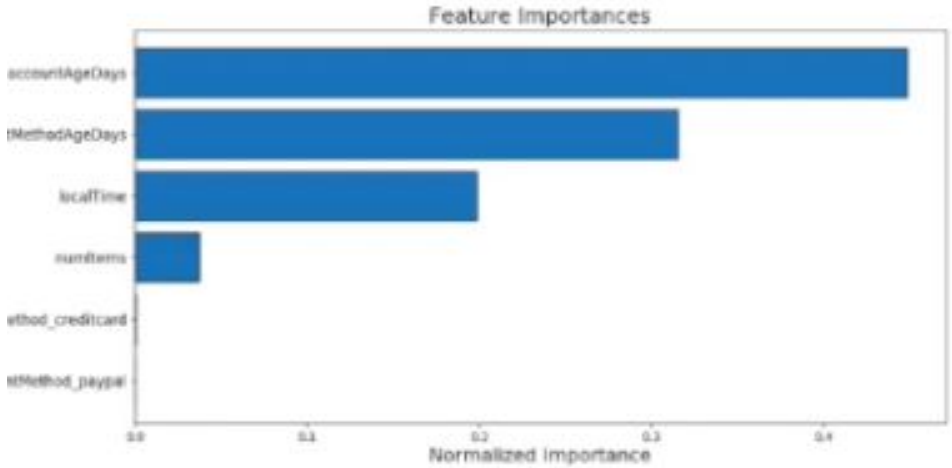
The `sklearn.feature_selection` module implements feature selection algorithms. It currently includes univariate filter selection methods and the recursive feature elimination algorithm.

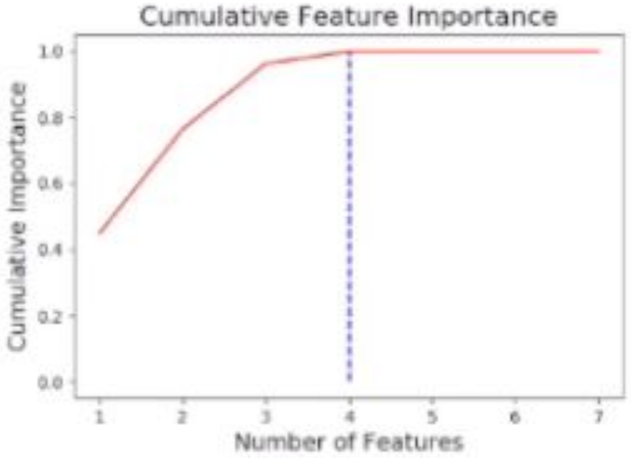
User guide: See the Feature selection section for further details.

`feature_selection.GenericUnivariateSelect` ([…])	Univariate feature selector with configurable strategy.
`feature_selection.SelectPercentile` ([…])	Select features according to a percentile of the highest scores.
`feature_selection.SelectKBest` ([score_func, k])	Select features according to the k highest scores.
`feature_selection.SelectFpr` ([score_func, alpha])	Filter: Select the pvalues below alpha based on a FPR test.
`feature_selection.SelectFdr` ([score_func, alpha])	Filter: Select the p-values for an estimated false discovery rate
`feature_selection.SelectFromModel` (estimator)	Meta-transformer for selecting features based on importance weights.
`feature_selection.SelectFwe` ([score_func, alpha])	Filter: Select the p-values corresponding to Family-wise error rate
`feature_selection.RFE` (estimator[, …])	Feature ranking with recursive feature elimination.
`feature_selection.RFECV` (estimator[, step, …])	Feature ranking with recursive feature elimination and cross-validated selection of the best number of features.
`feature_selection.VarianceThreshold` ([threshold])	Feature selector that removes all low-variance features.
`feature_selection.chi2` (X, y)	Compute chi-squared stats between each non-negative feature and class.
`feature_selection.f_classif` (X, y)	Compute the ANOVA F-value for the provided sample.
`feature_selection.f_regression` (X, y[, center])	Univariate linear regression tests.
`feature_selection.mutual_info_classif` (X, y)	Estimate mutual information for a discrete target variable.
`feature_selection.mutual_info_regression` (X, y)	Estimate mutual information for a continuous target variable.

Detección de fraude

```python
fs.identify_zero_importance(task = 'classification', eval_metric = 'auc',
                            n_iterations = 10, early_stopping = False)

fs.plot_feature_importances(threshold = 0.99, plot_n = 12)

print(fs.feature_importances.head(10))
```





feature	importance	normalized_importance	cumulative_importance
accountAgeDays	1101.0	0.448838	0.448838
paymentMethodAgeDays	773.0	0.315124	0.763962
localTime	487.0	0.198532	0.962495
numItems	91.0	0.037097	0.999592
paymentMethod_creditcard	1.0	0.000408	1.000000
paymentMethod_paypal	0.0	0.000000	1.000000
paymentMethod_storecredit	0.0	0.000000	1.000000

Detección de intrusiones

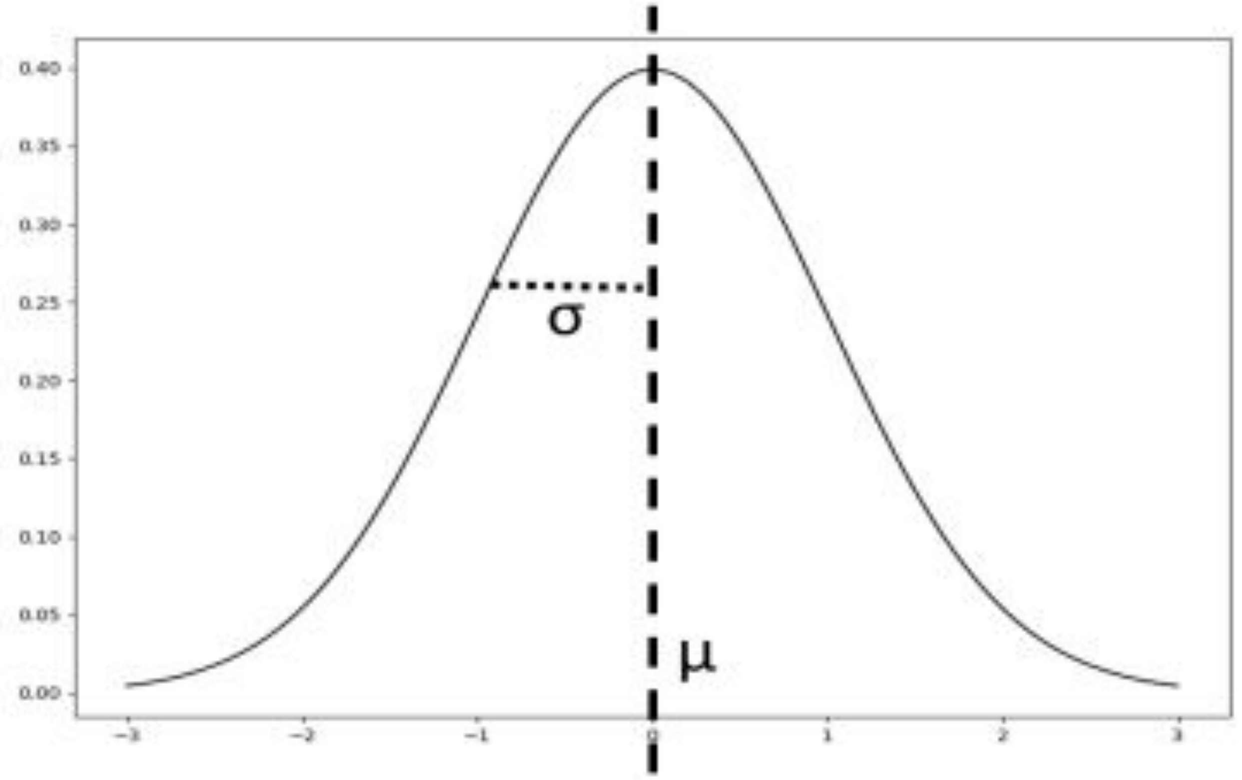
- **La detección de intrusiones se cataloga principalmente en dos categorías:**
  - **Basado en reglas y heurísticas: Genera un numero reducido de falsos positivos. Detecta ataques conocidos. No funciona correctamente para la detección de nuevos ataques.**
  - **Basado en anomalías: Perfila el comportamiento normal del sistema. Es capaz de detectar ataques nuevos. Puede generar un numero mayor de falsos positivos.**

Detección de anomalías

- ¿Cómo saber si hay una anomalía en su red?
  - Exfiltración de datos
  - Inicios de sesión atípicos
- Observar eventos anómalos es raro, por lo que los conjuntos de datos de anomalías son relativamente pequeños.
- Mejor ajuste: aprendizaje no supervisado

Distribución Gaussiana

$$\phi_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathbb{R}.$$



Distribución Gaussiana

- 1. Seleccionar características que pueden determinar que un ejemplo sea anómalo.
- 2. Ajustamos los parámetros del modelo.
  - Se calculan los parámetros para cada una de las características
- 3. Dado un nuevo ejemplo, computamos la probabilidad p(x)
- 4. Si p(x) < epsilon*, lo consideramos una anomalía

Distribución Gaussiana

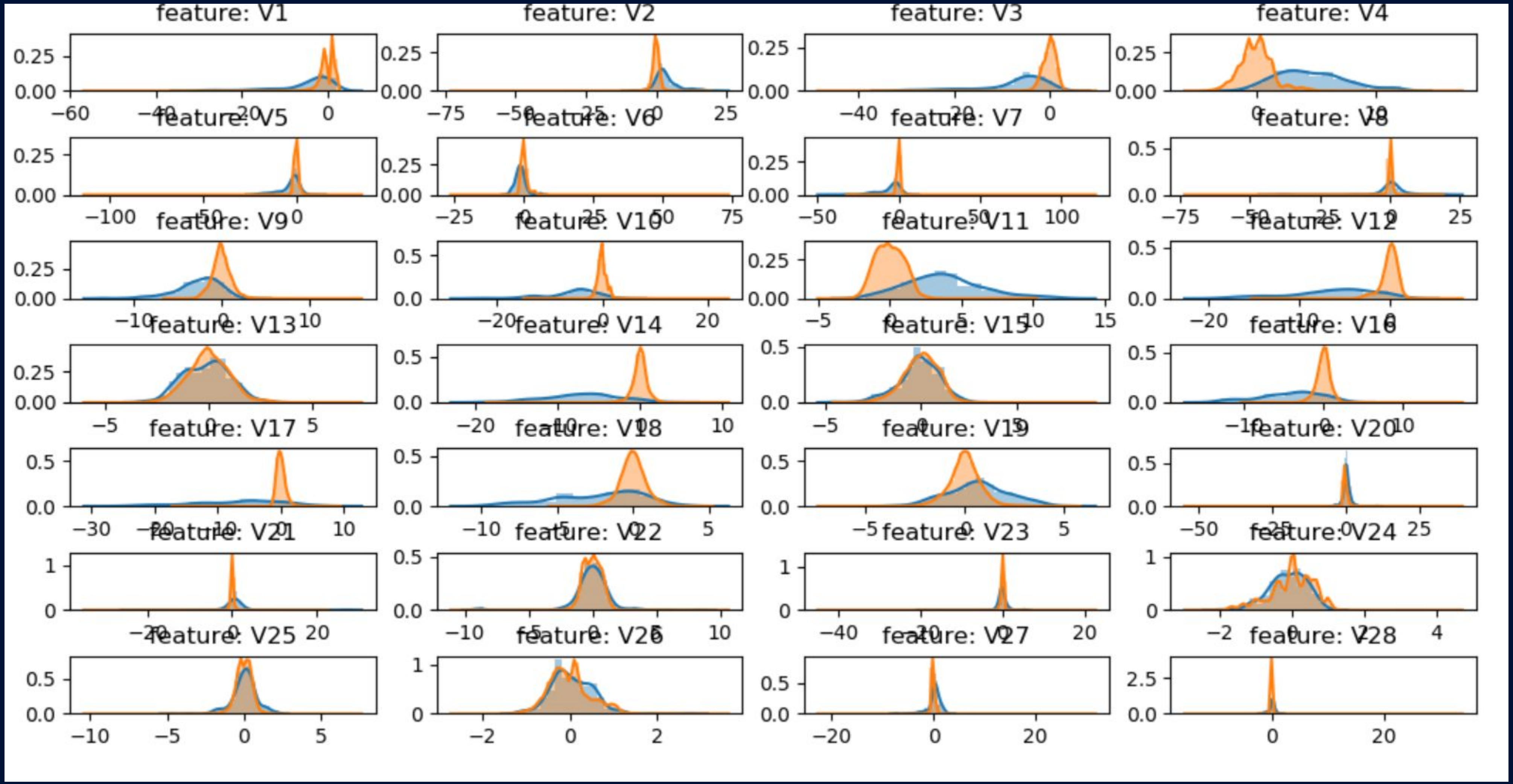
feature_importances_ ¶

Return the feature importances (the higher, the more important the feature).

Returns:	**feature_importances_** : ***array, shape = [n_features]*** The values of this array sum to 1, unless all trees are single node trees consisting of only the root node, in which case it will be an array of zeros.

```python
import pandas as pd
from sklearn.ensemble import RandomForestClassifier

df = pd.read_csv("creditcard.csv")

features = [f for f in list(df)
            if f not in ["Class", "Amount", "Time"]]

rnd_clf = RandomForestClassifier(n_estimators = 100,
                                 criterion = 'entropy',
                                 random_state = 0)

rnd_clf.fit(df.iloc[:,1:29],df.iloc[:,30])

x, y = (list(x) for x in zip(*sorted(zip(rnd_clf.feature_importances_, df.iloc[:,1:29].columns), reverse = True)))

for xe, ye in zip(x, y):
    print(ye, "-", xe)
```

Distribución Gaussiana



Distribución Gaussiana

```python
def estimateGaussian(dataset):
    mu = np.mean(dataset, axis=0)
    sigma = np.cov(dataset.T)
    return mu, sigma

def multivariateGaussian(dataset,mu,sigma):
    p = multivariate_normal(mean=mu, cov=sigma)
    return p.pdf(dataset)

def select_threshold(p_val, y_val):
    best_f1 = 0
    best_ep = 0
    ep = 1e-100
    print()
    for i in range(2000):
        print("\rSearching the best threshold {0}%".format(
            int((i + 1) / 2000 * 100)), end='')
        ep = ep*1.1
        predictions = (p_val < ep)
        f = f1_score(y_val, predictions, average='binary')
        if f > best_f1:
            best_f1 = f
            best_ep = ep
    return (best_f1, best_ep)
```

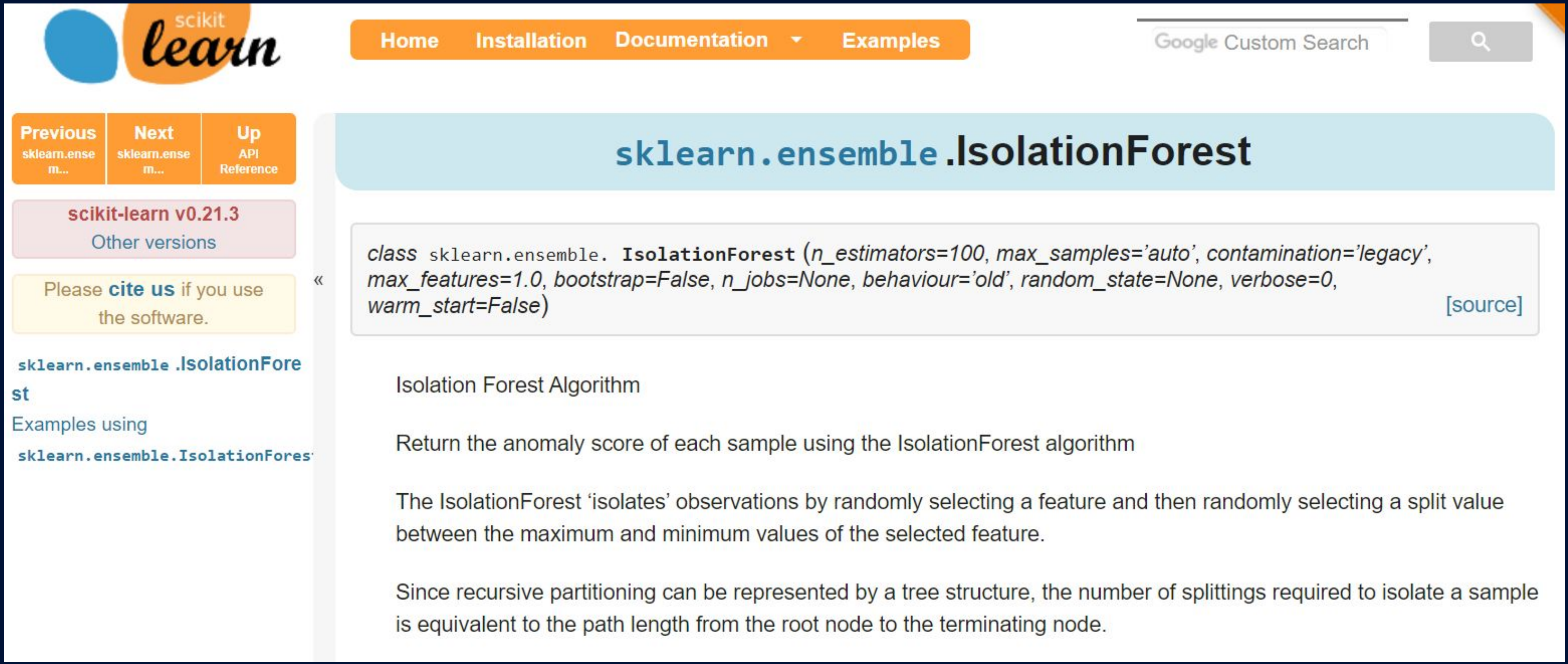
Distribución Gaussiana

```
df = df[['V17','V14','V12','V10','V11','V16','V4','V3','V7','V9','V18', 'Class']]

# Diviendo el conjunto de datos
df_anom = df[df.Class == 1]
df_norm = df[df.Class == 0]

# Conjunto de entrenamiento 60%
X_train = df_norm.sample(frac=0.6)
X_train.drop("Class", axis=1, inplace=True)

# Conjunto de validacion y pruebas 40%
X_val_test = df_norm[~df_norm.index.isin(X_train.index)]

# Conjunto validacion 20%, pruebas 20%
X_val_norm = X_val_test.sample(frac=0.5)
X_test_norm = X_val_test[~X_val_test.index.isin(X_val_norm.index)]

# Ejemplos anomalos 50% test, 50% pruebas
X_val_anom = df_anom.sample(frac=0.5)
X_test_anom = df_anom[~df_anom.index.isin(X_val_anom.index)]

# Juntamos ejemplos normales y anomalos en validacion y test
X_val = pd.concat([X_val_norm, X_val_anom], ignore_index=True)
y_val = X_val.Class
X_val.drop("Class", axis=1, inplace=True)

X_test = pd.concat([X_test_norm, X_test_anom], ignore_index=True)
y_test = X_test.Class
X_test.drop("Class", axis=1, inplace=True)
```

Distribución Gaussiana

```
mu, sigma = estimateGaussian(X_train)
p = multivariateGaussian(X_train,mu,sigma)
p_val = multivariateGaussian(X_val,mu,sigma)
p_test = multivariateGaussian(X_test,mu,sigma)

bestf1_val, epsilon = select_threshold(p_val, y_val)

predictions = (p_test < epsilon)

print("\n\nBest f1_score in validation:", bestf1_val)
print("Best epsilon in validation:", epsilon)
F1score = f1_score(y_test, predictions, average = "binary")
print ('\nF1 Score in testing: %f' %F1score)
```

Isolation forest



scikit learn

Home | Installation | Documentation | Examples

Google Custom Search

Previous sklearn.ensem... | Next sklearn.ensem... | Up API Reference

scikit-learn v0.21.3
Other versions

Please **cite us** if you use the software.

sklearn.ensemble.IsolationForest
Examples using sklearn.ensemble.IsolationFores

sklearn.ensemble.IsolationForest

class `sklearn.ensemble.` **`IsolationForest`** (*n_estimators=100, max_samples='auto', contamination='legacy', max_features=1.0, bootstrap=False, n_jobs=None, behaviour='old', random_state=None, verbose=0, warm_start=False*) [source]

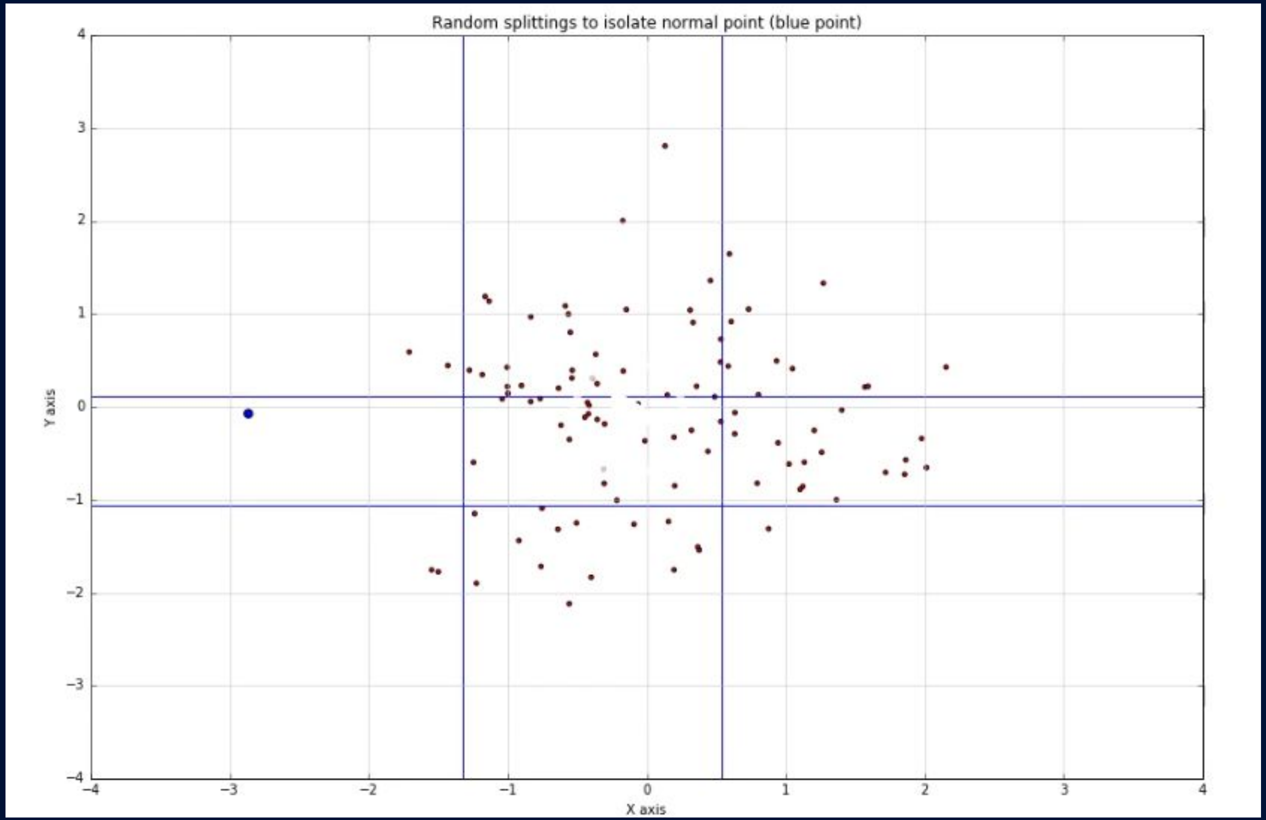
Isolation Forest Algorithm

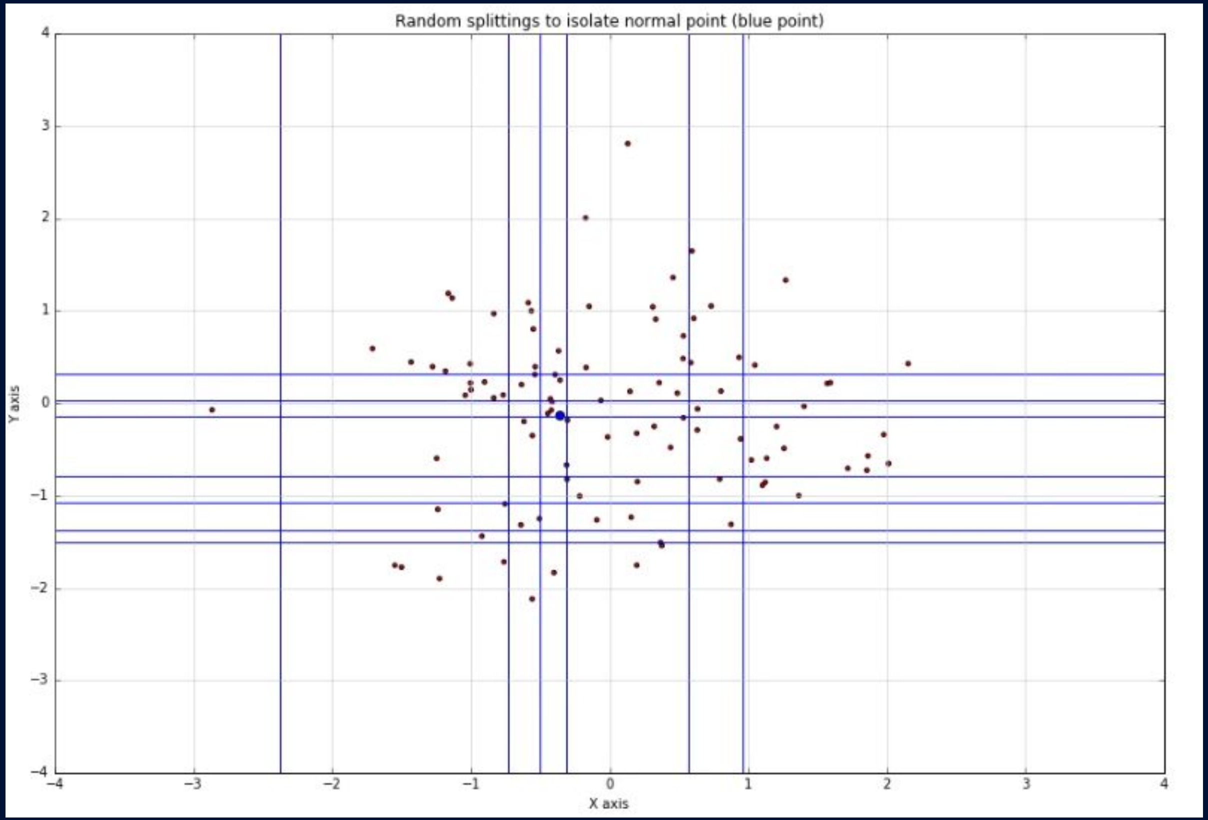
Return the anomaly score of each sample using the IsolationForest algorithm

The IsolationForest 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

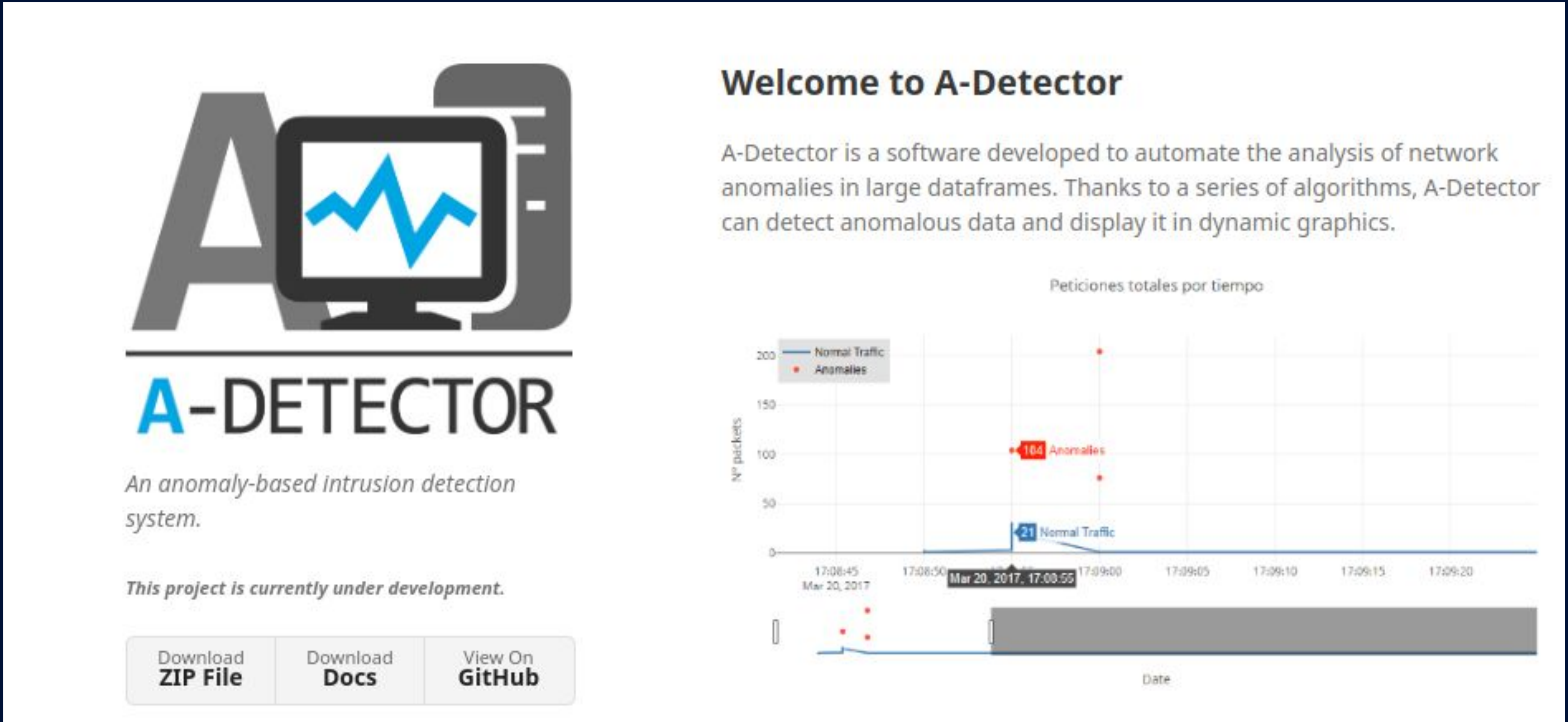
Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

Isolation forest

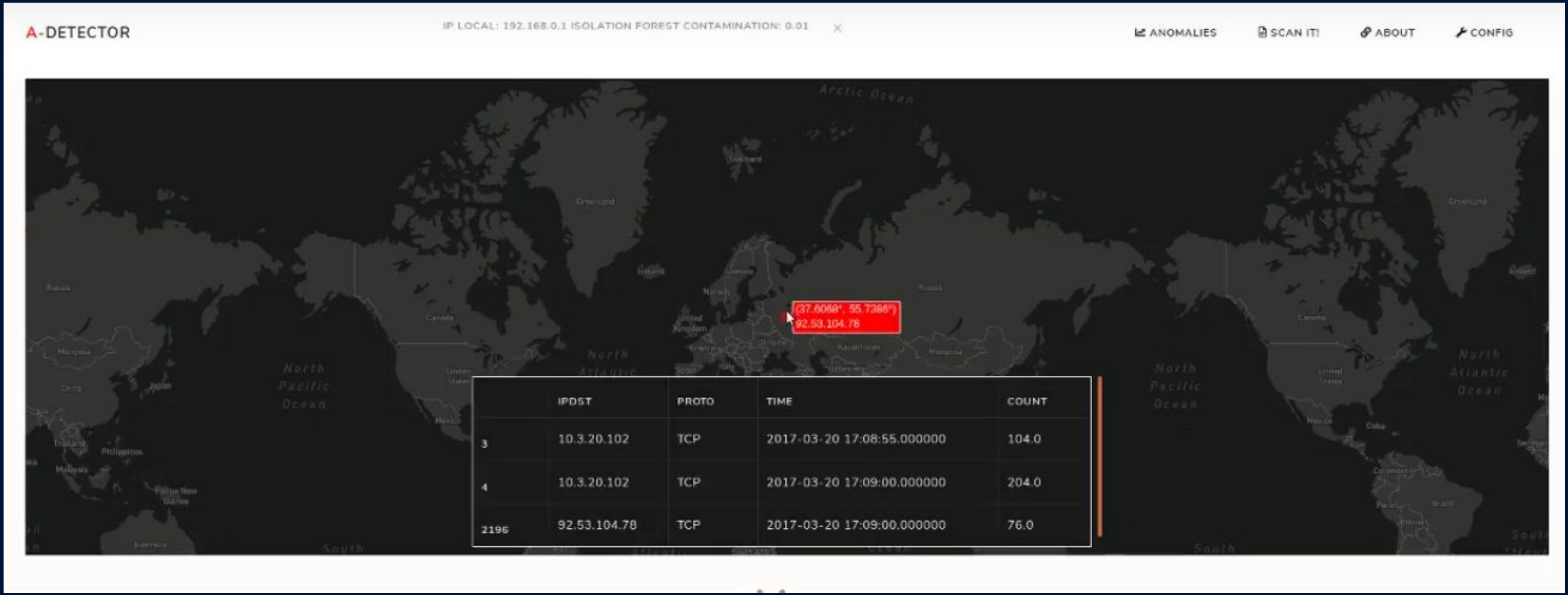




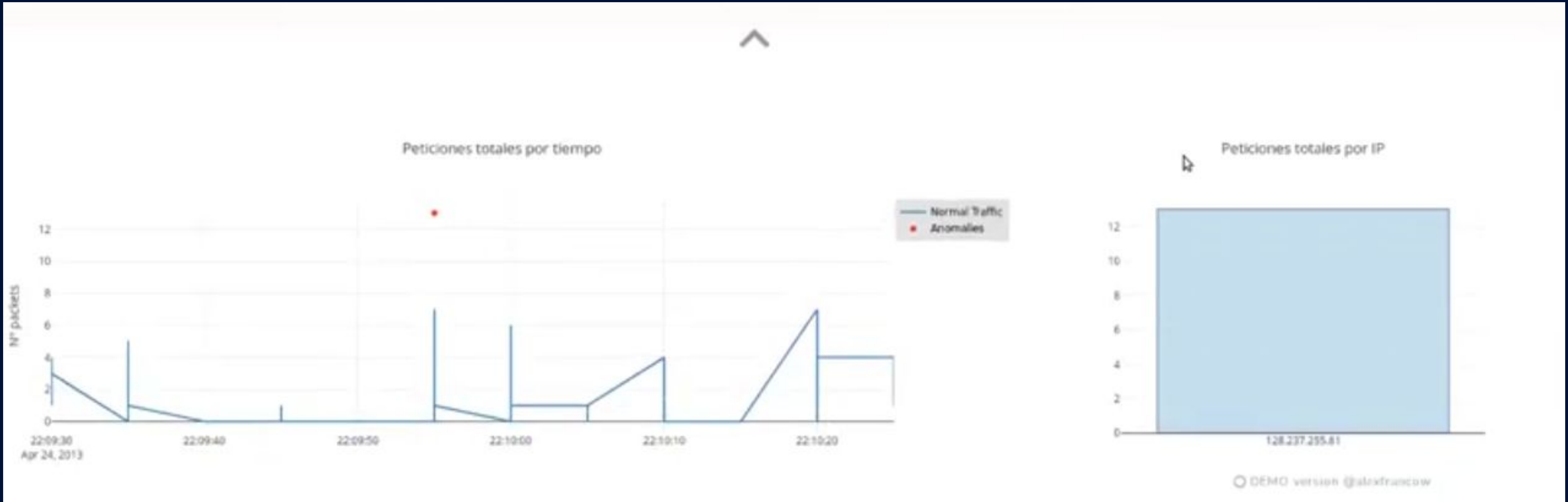
A-Detector: un IDS basado en anomalías



A-Detector: un IDS basado en anomalías



A-Detector: un IDS basado en anomalías



Repositorios con ejemplos

https://github.com/bschieche/python-anomaly-detection

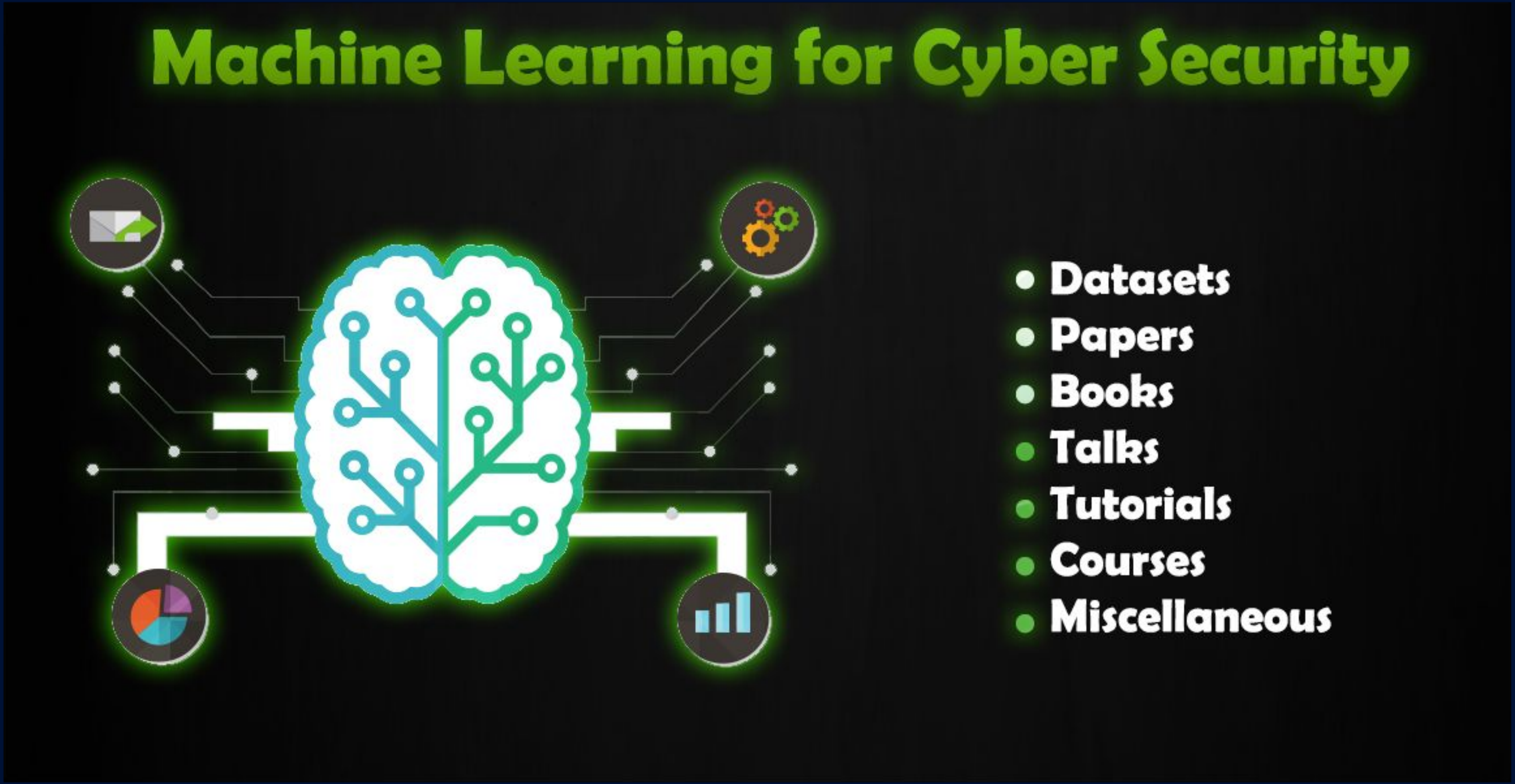
https://nbviewer.jupyter.org/github/bschieche/python-anomaly-detection/blob/master/anomaly_detection.ipynb

https://github.com/albertcthomas/anomaly_detection_lab

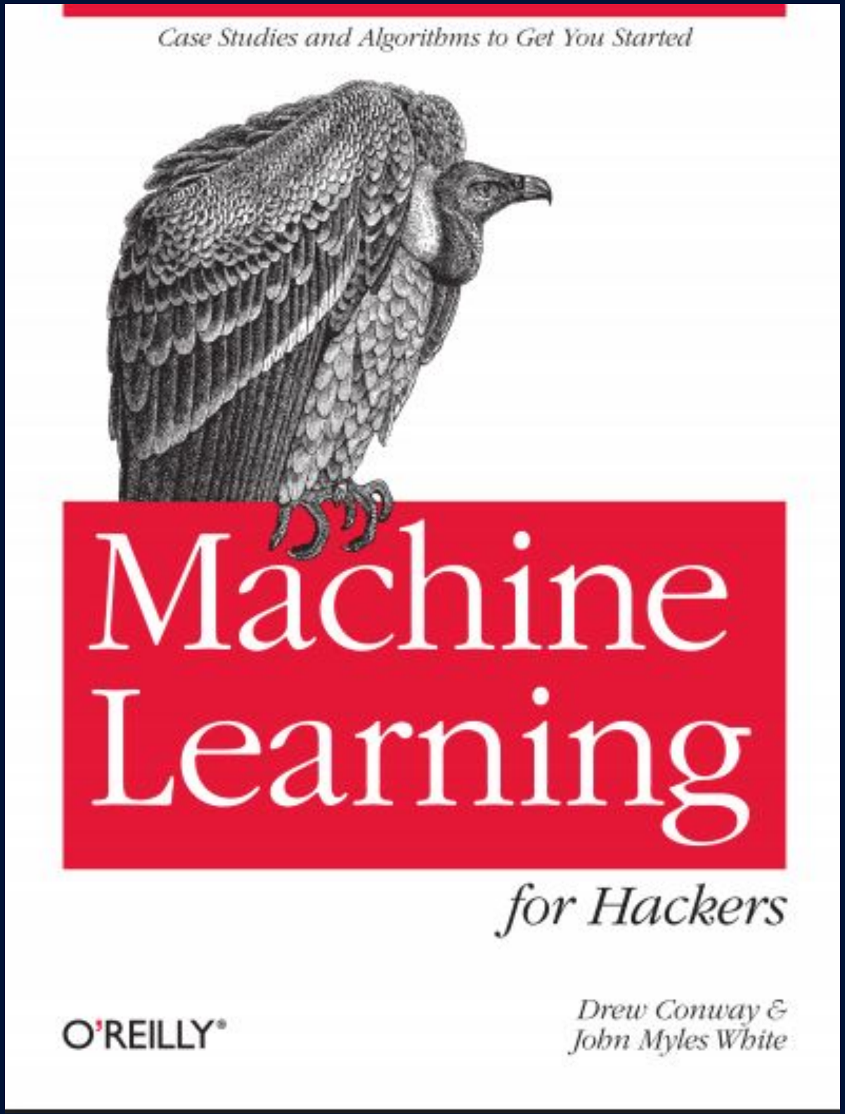
https://github.com/slrbl/Intrusion-and-anomaly-detection-with-machine-learning

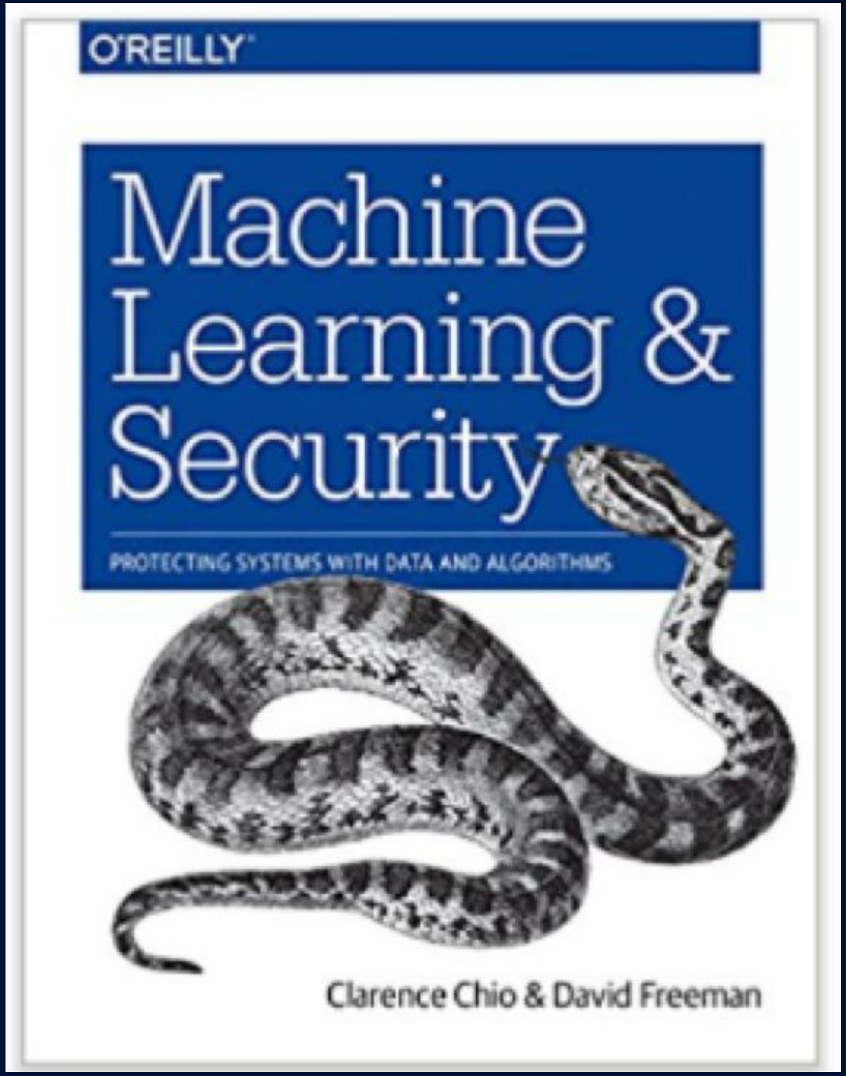
Conclusiones

Resources

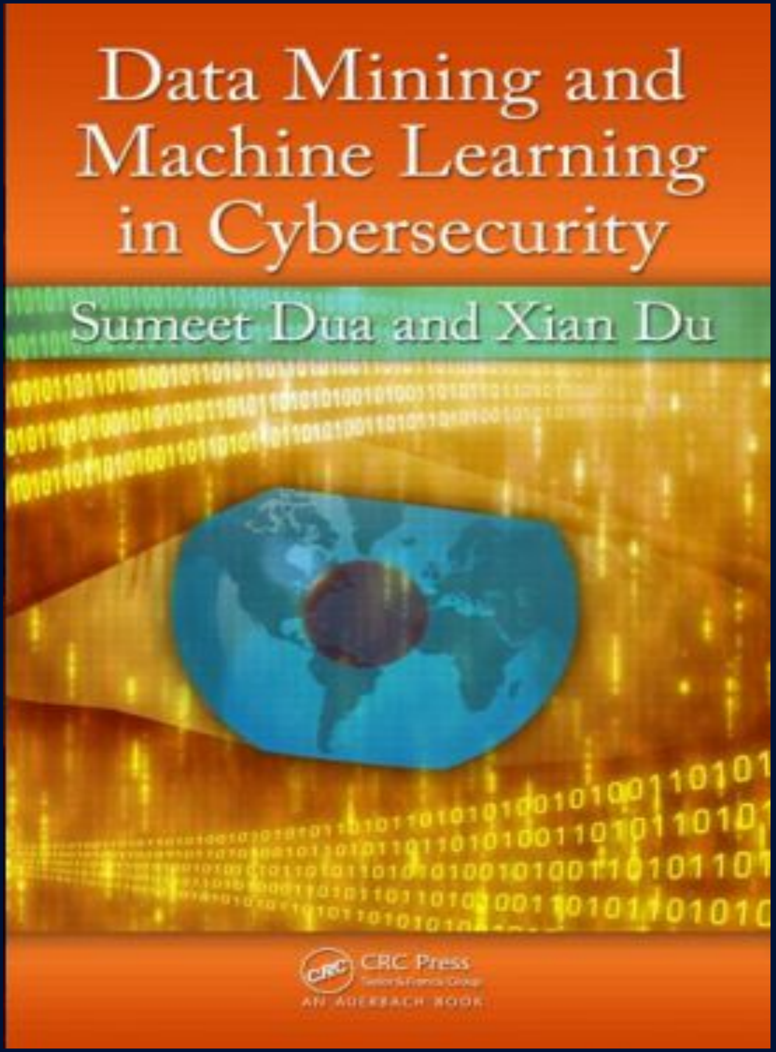

Machine Learning for Cyber Security
Datasets
Papers
Books
Talks
Tutorials
Courses
Miscellaneous

Resources

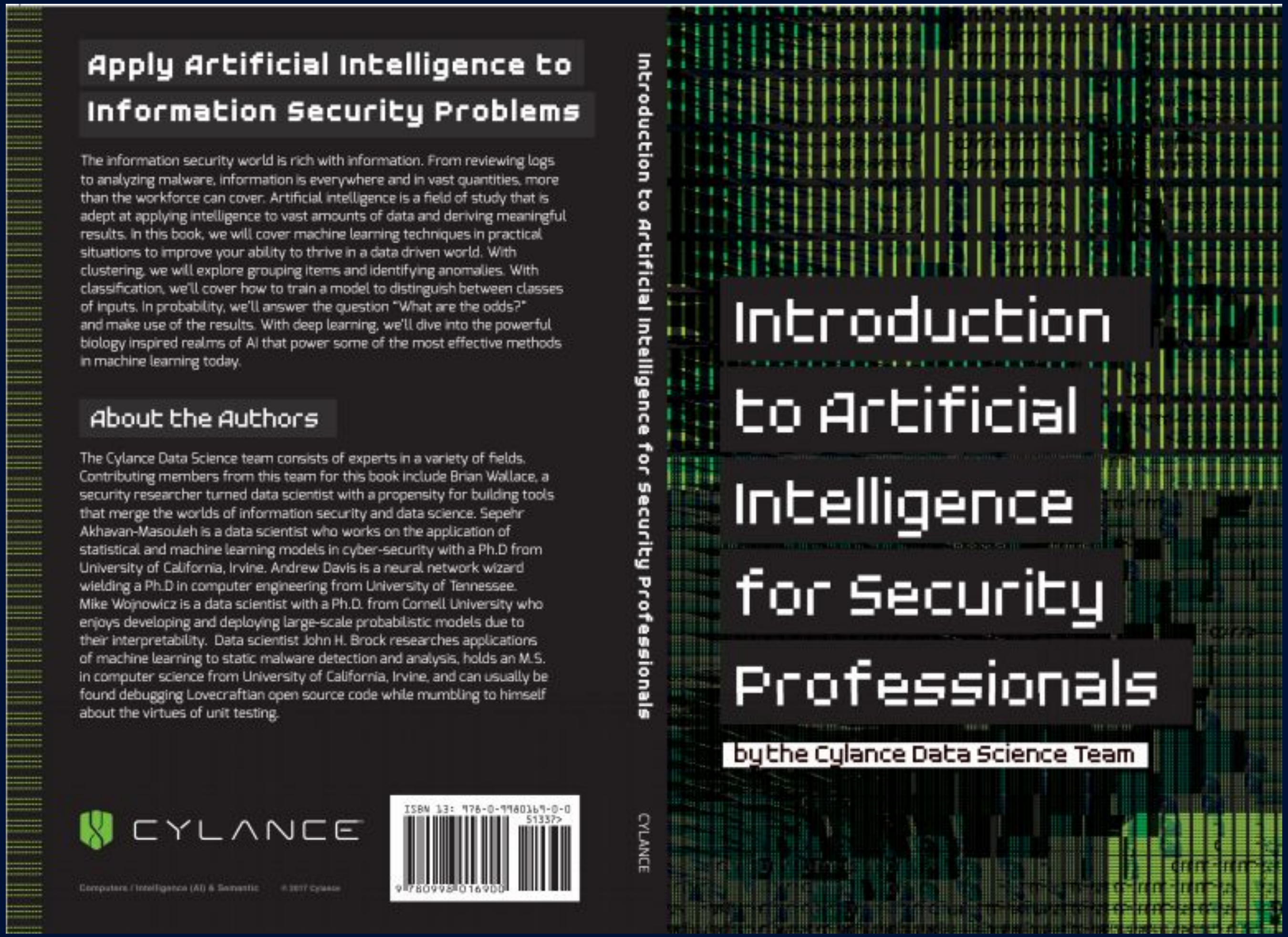

Case Studies and Algorithms to Get You Started
Machine
Learning
for Hackers
O'REILLY
Drew Conway &
John Myles White


O'REILLY
Machine
Learning &
Security
PROTECTING SYSTEMS WITH DATA AND ALGORITHMS
Clarence Chio & David Freeman

Resources

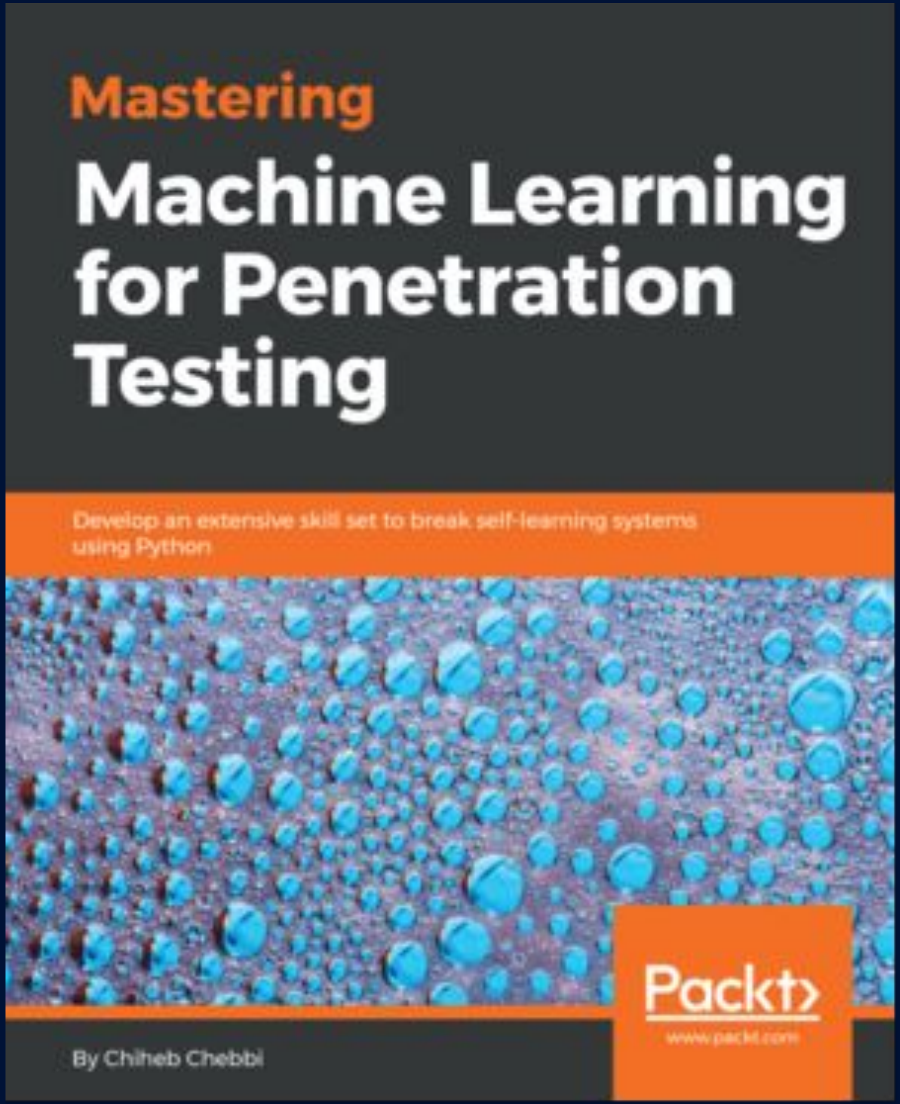


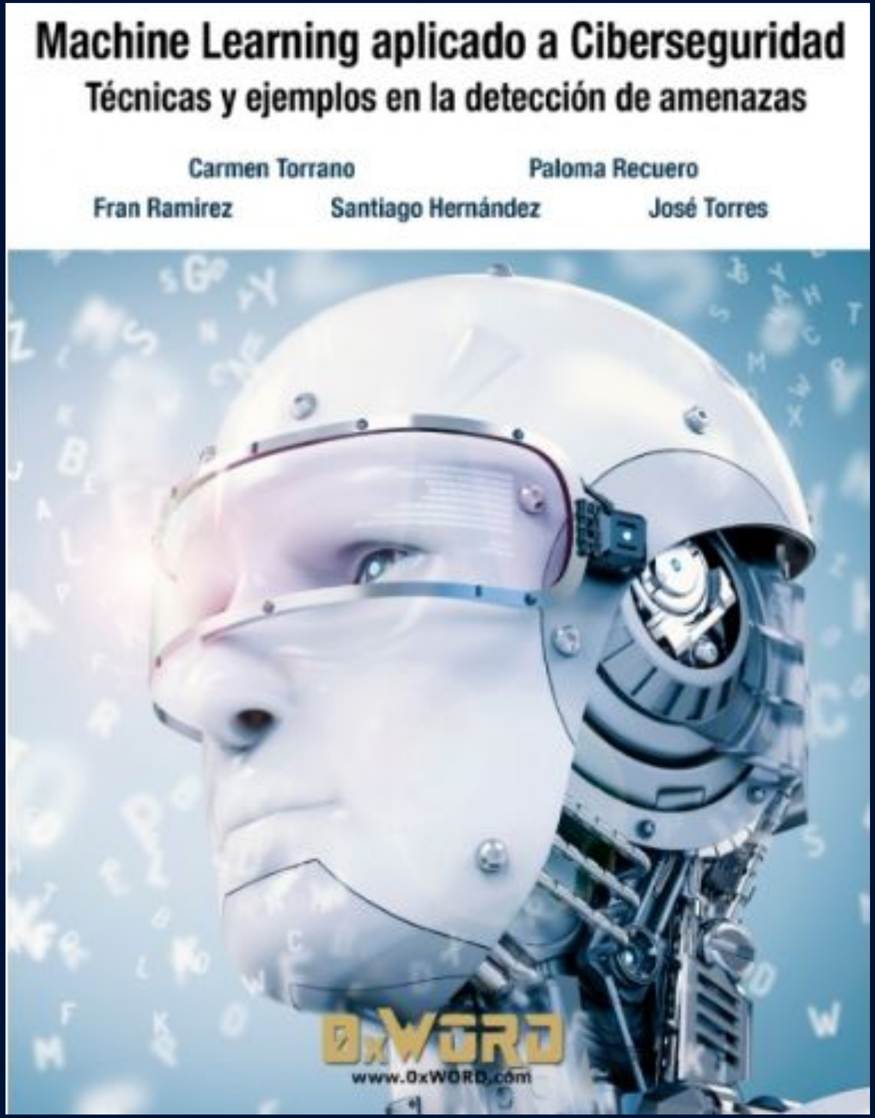
Data Mining and Machine Learning in Cybersecurity
Sumeet Dua and Xian Du
CRC Press



Apply Artificial Intelligence to Information Security Problems
The information security world is rich with information. From reviewing logs to analyzing malware, information is everywhere and in vast quantities, more than the workforce can cover. Artificial intelligence is a field of study that is adept at applying intelligence to vast amounts of data and deriving meaningful results. In this book, we will cover machine learning techniques in practical situations to improve your ability to thrive in a data driven world. With clustering, we will explore grouping items and identifying anomalies. With classification, we'll cover how to train a model to distinguish between classes of inputs. In probability, we'll answer the question "What are the odds?" and make use of the results. With deep learning, we'll dive into the powerful biology inspired realms of AI that power some of the most effective methods in machine learning today.
About the Authors
The Cylance Data Science team consists of experts in a variety of fields. Contributing members from this team for this book include Brian Wallace, a security researcher turned data scientist with a propensity for building tools that merge the worlds of information security and data science. Sepehr Akhavan-Masouleh is a data scientist who works on the application of statistical and machine learning models in cyber-security with a Ph.D from University of California, Irvine. Andrew Davis is a neural network wizard wielding a Ph.D in computer engineering from University of Tennessee. Mike Wojnowicz is a data scientist with a Ph.D. from Cornell University who enjoys developing and deploying large-scale probabilistic models due to their interpretability. Data scientist John H. Brock researches applications of machine learning to static malware detection and analysis, holds an M.S. in computer science from University of California, Irvine, and can usually be found debugging Lovecraftian open source code while mumbling to himself about the virtues of unit testing.
CYLANCE
Introduction to Artificial Intelligence for Security Professionals
CYLANCE
Introduction to Artificial Intelligence for Security Professionals
by the Cylance Data Science Team

Resources


Mastering
Machine Learning for Penetration Testing
Develop an extensive skill set to break self-learning systems using Python
Packt>
www.packt.com
By Chiheb Chebbi


Machine Learning aplicado a Ciberseguridad
Técnicas y ejemplos en la detección de amenazas
Carmen Torrano
Paloma Recuero
Fran Ramirez
Santiago Hernández
José Torres
0xWORD
www.0xWORD.com

Resources

- https://towardsdatascience.com/machine-learning-for-cybersecurity-101-7822b802790b

PY CON ES

ALICANTE 2019