Taller de Aprendizaje Automático

Actividades Taller 4

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Montevideo, 2024

Tabla de contenido

Objetivos del Taller 4

- Abordar un problema de detección de anomalías, y ver las diferencias con un problema de clasificación convencional.
- Trabajar con algoritmos de aprendizaje no supervisado.
- Crear detectores compatibles con los **pipelines** de *scikit-learn*.

El problema y los datos

- Se quieren detectar *conexiones potencialmente peligrosas* en redes de computadoras a partir de datos de tráfico
- Se trabajará con el conjunto KDD Cup'99
  - Los ataques se clasifican en cuatro categorías:
    - DOS *denial-of-service*: Disminución o pérdida total del servicio.
    - R2L *remote-to-local*: acceso remoto no autorizado . Ej: adivinar contraseña
    - U2R *user-to-root*: acceso no autorizado a privilegios de superuser (root)
    - Probing: sondeo, vigilancia. Ej: escaneo de puertos

Tipo	Ataque
DOS	back, ~~land~~, neptune, pod, smurf, teardrop
R2L	~~ftp_write~~, guess_passwd, ~~imap~~, ~~multihop~~, ~~phf~~, ~~spy~~, warezclient, ~~warezmaster~~
U2R	~~buffer_overflow~~, ~~loadmodule~~, ~~perl~~, rootkit
probing	ipsweep, nmap, portsweep, satan

Table: Tipos de ataques presentes en el conjunto de entrenamiento original

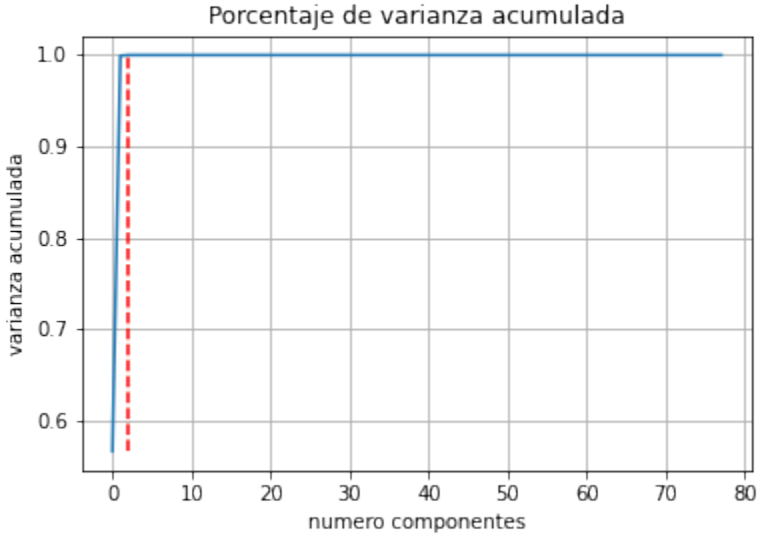
Abordaje

- Se abordará el problema como uno de detección de anomalías
- Se evaluarán distintas formas de modelar la normalidad:
  - PCA
  - K-Means
  - Mezcla de Gaussianas

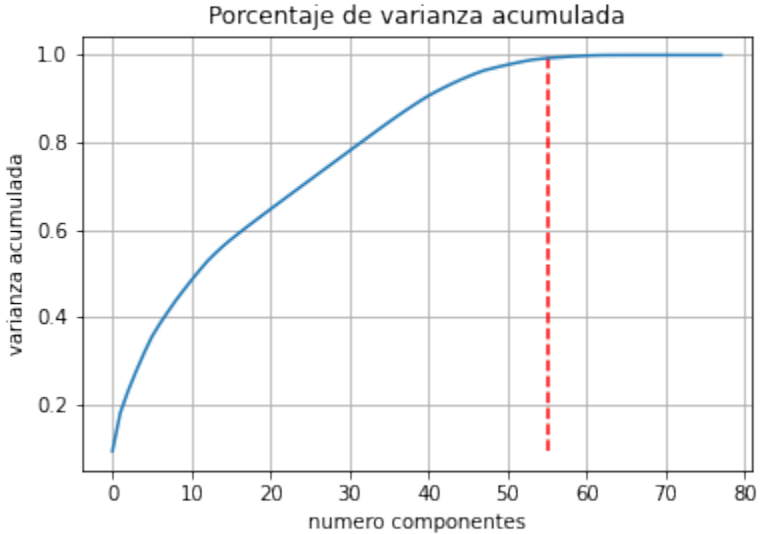
Para hacer ahora

- Levantar los datos
- Mirar las características y sus codificaciones
- Ver la relación entre la cantidad de datos normales y ataques
- *¿Por qué tiene sentido encarar este problema como uno de detección de anomalías?*
- Separar 90000 muestras "normales" para modelar la clase normal y reservar el resto para determinar el punto de funcionamiento
- Generar un pipeline de preprocesamiento
- Hacer PCA sobre los datos de entrenamiento y graficar como varía la varianza acumulada en función de la cantidad de componentes

PCA


Porcentaje de varianza acumulada
varianza acumulada
1.0
0.9
0.8
0.7
0.6
0
10
20
30
40
50
60
70
80
numero componentes

PCA


Porcentaje de varianza acumulada
varianza acumulada
numero componentes
1.0
0.8
0.6
0.4
0.2
0
10
20
30
40
50
60
70
80

Abordaje mediante PCA

- Se encuentran las componentes principales del conjunto 'normal' de forma de mantener el 99% de la varianza
- Dado un nuevo dato:
  1. Se lo proyecta utilizando las componentes principales
  2. Se reconstruye utilizando las componentes principales
  3. Se calcula el error de reconstrucción y se decide

Implementación usando PCA

```python
from sklearn.base import BaseEstimator, OutlierMixin
from sklearn.utils.validation import check_array, check_is_fitted
from sklearn.decomposition import PCA

class AD_PCA(BaseEstimator, OutlierMixin):
    def __init__(self, n_comp=None):
        self.n_comp = n_comp

    def fit(self, X, y=None):

        self.X = X
        self.y = y

        # Agregar código---

        #------------------
        return self

    def score(self, X, y=None):
        # Se verifica que los datos sean válidos
        X = check_array(X)
        # Se verifica que el modelo haya sido entrenado
        check_is_fitted(self, ['X', 'y'])

        # Agregar código---

        #------------------
        return score
```

Abordaje mediante K-Means

- Se encuentran los clusters (Se sugiere ver *Finding the optimal number of clusters del Capítulo 9*)
- Dado un nuevo dato:
  1. Se mide la cercanía a todos los clusters
  2. Se define en función de la parte anterior

Abordaje mediante Mezcla de Gaussianas

- Se encuentran la cantidad de mezclas (Se sugiere ver *Anomaly Detection Using Gaussian Mixture del Capítulo 9*)
- Dado un nuevo dato:
  1. Se calcula la log-verosimilitud
  2. Se define en función de la parte anterior