San Francisco crime: deep analysis and prediction model

Álvaro·López·Sánchez  
Julio·Sorroche·García  
Lupicinio·García·Ortiz

Abstract

En este proyecto utilizaremos la información de la actividades criminales producidas en la ciudad de San Francisco organizadas en un volúmen masivo de datos, y con el fin de seguir una serie de etapas de preprocesamiento, análisis para finalmente obtener un modelo de predicción. Para ello seguiremos diferentes perspectivas y metodologías a lo largo del almacenamiento, preprocesamiento y representación de la información con ayuda de MongoDb, Cassandra y Neo4j. Para estar mejor preparado para responder a cualquier actividad delictiva y predecirlos, es importante comprender los patrones para que se produzca un delito. Mediante varios modelos de clasificación y predicción basado en aprendizaje profundo supervisado y no supervisado; optaremos por la opción que se ajuste mejor para realizar predicciones.

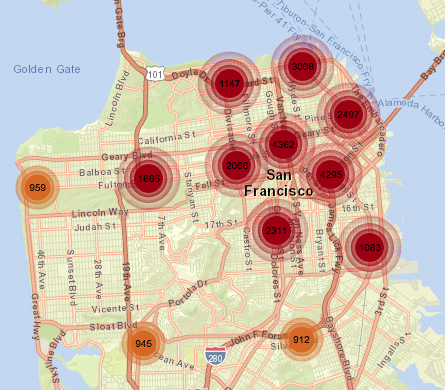
# Introducción

El objetivo principal del proyecto es clasificar la información de la actividad criminal de San Francisco dado el tiempo y la localización. Realizar un análisis y obtener un conjunto de modelos de representación de la información de éstas actividades; y que a partir de dichos modelos podamos consultar información con mejor enfoque y con valores más discretos. Para ello introduciremos las tecnologías MongoDb, Cassandra y Neo4j y obtener diferentes perspectivas de proceso y manejo de la información de forma respectiva.

En el resultado final la información disponible no solo se basa de la información en bruto obtenida a partir de un volumen masivo de datos correspondientes a incidencias desde el 2013 hasta la actualidad; sino que ademas al ajustar un modelo de predicción se podrá encontrar datos futuros; y la mejor manera es visualizar dichas predicciones sobre la información presente.

## I. Dataset

Nuestro dataset está publicamente disponible desde el portal del departamento Policial de *San Francisco*. Contiene datos de todas las incidencias generadas por actividades criminales producidas desde el año 2003 hasta la actualidad. De forma concreta, esta información se encuentra expuesta de forma pública desde el sistema de actualización diaria del *SFPD Crime Incident Reporting* (sistema de reportes del departamento policial) a traves de su plataforma Socrata.



La información esta representada por un conjunto de 2 millones de filas, que contiene información detallada de actividades criminales (o incidencias) representada por: *tipo de delito, resolución, distrito, coordenadas y dirección de la zona, nº de incidencia vinculada*; todos producidos entre el 2003 hasta la actualidad.

* **Dates -** fecha y hora del incidente del crimen
* **Category** - categoría del incidente criminal
* **Descript** - descripción detallada del incidente del crimen
* **Dayoftheweek** de la semana: el día si la semana
* **PdDistrict**- nombre del distrito del departamento de policía
* **Resolution**: cómo se resolvió el incidente del crimen
* **Address**: dirección aproximada del incidente delictivo
* X - longitud
* Y - latitud

En el caso de las **coordenadas** (x,y). La X y la Y dan esencialmente el parámetro de ubicación. Es un dato interesante que veremos aprovechar mediante tecnologia que proporciona **MongoDb** con respecto a geolocalizaciones.

La fecha se agrupa como una fecha, hora y día de la semana. Esto se divide para su uso posterior en las siguientes secciones.

150060275;NON-CRIMINAL;LOST PROPERTY;Monday;2015-01-19 14:00:00;19;2015;01;14;MISSION;NONE;18TH ST / VALENCIA ST;-122.42158168137;37.7617007179518;(37.7617007179518, -122.42158168137);15006027571000

150098210;ROBBERY;ROBBERY, BODILY FORCE;Sunday;2015-02-01 15:45:00;01;2015;02;15;TENDERLOIN;NONE;300 Block of LEAVENWORTH ST;-122.414406029855;37.7841907151119;(37.7841907151119, -122.414406029855);15009821003074

## II. Fases

A partir de la estructura de datos definida para el dataset se llevará a cabo una serie de procesos y fases, que nos permitan almacenar dicha información desde distintas perspectivas de almacenaje y representación masiva de datos

* **Mongodb**, como manejo de información a través de documentos
* **Cassandra**, con potencial residente en la arquitectura para el volumen de datos
* **Neo4j**, como orientación en la busqueda de relaciones entre los datos

# Contextualización del proyecto

## I. Antecedentes

.....

## II. Proyectos en la actualidad

## .....

# Fundamentos teóricos/prácticos

En este apartado inicialmente se discutiran las capacidades en las fases de preprocesamiento, análisis y estructura de la información de cada tecnología, conectores usados para realizar la representación de la información.

## Estructura y arquitectura de la información

En este apartado se realiza un desglose teórico de como cada modelo de base de datos correspondiente a las tecnologias usadas dispone la información, y de como la arquitectura se basa.

[I. Cassandra](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/cassandra/readme.md#preprocesamiento-e-importaci%C3%B3n-de-datos)

II. [Neo4j](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/neo4j/readme.md#preprocesamiento-e-importaci%C3%B3n-de-datos)

III. [Mongodb](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/mongodb/readme.md#preprocesamiento-e-importaci%C3%B3n-de-datos)

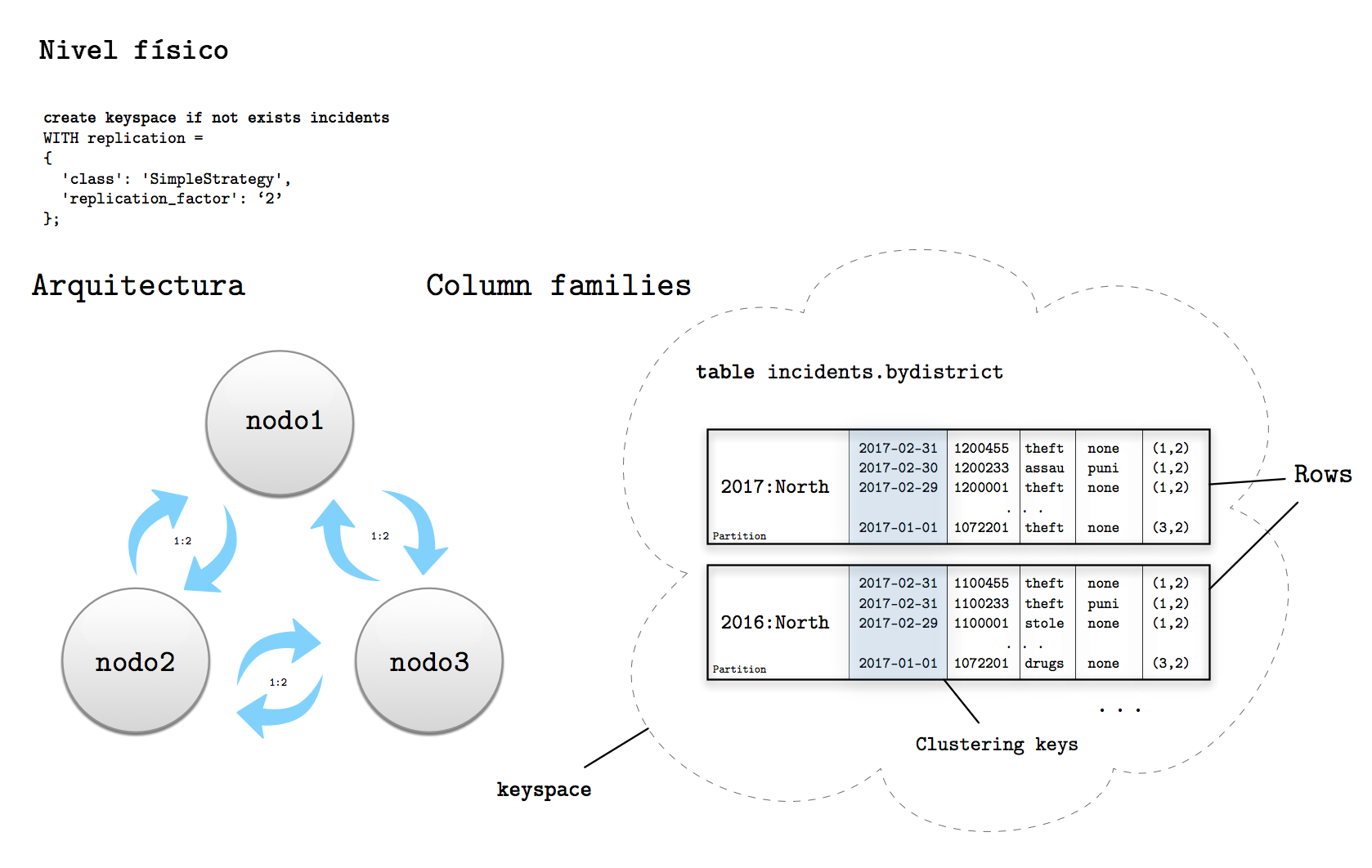
### I. Cassandra

Cassandra es una base de datos dependiente del caso de uso. En la mayoría de los casos, una simple instancia de MySQL o PostgreSQL haría mejor el trabajo.

La idea principal es que encontremos que Cassandra pueda ofrecer facilidades con respecto a la organización de los **atributos** en nuestro modelo de datos.

#### Estructura de la información

Unas de las características principales de Cassandra es la separación entre nivel físico y lógico.

[](https://github.com/orial/grupo_Spark/blob/master/Procesamiento/docs/cassandra/nivel_fisico.png)

* **Column families** son abstracciones lógicas unitarias que conforman las tablas fisicas de CQL. Es la forma en la que disponemos la información, el esquema de organización de los campos y valores, mediante distintos tipos de columnas.
* **Particiones**. Define la estructura de compactación que engloban parte de la información de una tabla en concreto, y que las filas que la componen tienen en común el mismo criterio lógico dado por una/s clave/s de partición. Las filas que componen cada partición pueden estar ordenadas dependiendo del criterio de ordenación mediante definición de columnas como claves clustering.
* **Rows** Son las unidades de forman las column families y son instancias de las particiones. Una partición puede contener una gran cantidad de rows. Por cada partición, se pueden definir la ordenación de las filas que lo componen.

En nuestro caso, las particiones van a ser tipo **multi-row** ya que contendrán una cantidad masiva de filas ordenadas (por timestamp), y tendrán en común un determinado criterio, dado por una composite key (clave de partición compuesta): por distrito o por zona a parte del año, conformando claves de distinto tipo dependiendo del modelo de consulta que se requeiera por la aplicación:

* Por año: 2017
* Por zona/año: 2017:north
* Por tipo de delito/año: 2017:theft

La definición de las tablas necesarias para las consultas las podemos encontrar bajo el fichero: .

Con respecto a las ordenación de las filas las claves de ordenación o clustering han sido fijadas sobre las columnas claves para seleccionar filas y definir cierto criterio en la realización de cualquier consulta. Generalmente la columna time ha sido seleccionada en todas las opciones y de forma DESCENDENTE, con el fin de que a nivel físico sean ordenadas de más reciente a menos reciente; pues por usabilidad no es cuestionable dicho orden.

#### Arquitectura

A **nivel de arquitectura**, solo mencionar que los datos de las tablas son alojadas como **column families** y dicha información es compartida de forma redundante entre los nodos que compongan la red Cassandra y del grado de replicación de como se configure el keyspace

<...>

## II. Neo4j

**Neo4j** es una base de datos basada en grafos, utilizan una estructura de los lenguajes de programación conocida como *grafos*, que permite relacionar los datos a través de enlaces que facilitan el recorrido a través de ellos. Se usa para almacenar datos con relaciones complejas (por ejemplo, rutas con coordenadas GPS, relaciones sociales, ...)

### Estructura de la información

<...>

Arquitectura

<...>

## III. Mongodb

<...>

### Estructura de la información

<...>

Arquitectura

<...>

## Comparativa

En la siguiente rubrica realizamos comparaciones del tipo de arquitectura y de la capacidad de la estructuración de la información de las tres tecnologías.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | Ventajas | | Desventajas |
| Cassandra | * **Consistencia estable y replicación de datos**. La relación entre la **estructura lógica y la física** permite que la información quede estructurada y organizada entre los nodos, con el fin de optimizar y almacenar la información entre ellos. * **Redundancia de información**. La información se encuentra organizada entre los nodos de forma redundante a su vez. * CQL Lenguaje de consulta de Cassandra es una forma bastante familiar para hacer consultas sobre Cassandra. Es un subconjunto de SQL y tiene muchas de las mismas características, haciendo que la transición de un RDBMS basado en SQL a Cassandra sea menos discordante. | | * **Sin Consultas Ad-Hoc**: La capa de almacenamiento de datos de Cassandra es básicamente un sistema de almacenamiento de clave-valor. Esto significa que debe "modelar" sus datos en torno a las consultas que desea que surjan, en lugar de en torno a la estructura de los datos en sí. Esto puede llevar a almacenar los datos varias veces de diferentes maneras para poder satisfacer los requisitos de su aplicación. * **Sin agregaciones**, aunque en esta versión si las usaremos (3.11): las versiones más nuevas de Cassandra tendrán soporte limitado para agregaciones con una sola partición. **Esto es de uso muy limitado. Debido a que Cassandra es una tienda de valores clave, hacer cosas como SUM, MIN, MAX, AVG y otras agregaciones requieren una gran cantidad de recursos si es posible**. Si hacer un análisis ad-hoc es un requisito para su aplicación, entonces Cassandra puede no ser para usted. * CQL: Es fácil para alguien que proviene de SQL confundirse acerca de qué es o no compatible. Esto significa una frustración adicional (costos de lectura) para los desarrolladores que no conocen las limitaciones de Cassandra. | |
| Mongodb | <a rellenar> | | <a rellenar> | |
| Neo4j | <a rellenar> | | <a rellenar> | |

## Conexión a la estructura de la información: conectores

Actualmente, toda estructura de almacenamiento de datos posee algún tipo de software o interfaz de usuario oficial para el mantenimiento y gestión de la información que posee.

Esta aplicación, por lo general, está construida bajo uno de los lenguajes de programación comunes ¿Cómo consigue dicho lenguaje conectarse con la base de datos e interactuar con esta? Mediante el uso de driver y conectores. Hoy en día, el término “driver”, en este contexto, quiere referirse a ambas funcionalidades de conexión e interacción, mientras que conector es el módulo software que permite interacción. Esto implica que un conector está implícitamente contenido en un driver de bases de datos.

Dichos drivers nos permitirán por tanto recuperar la información almacenada mediante el lenguaje de consulta característico de cada sistema. Tener acceso a las funciones de los conectores nos ofrece la posibilidad de diseñar funciones auxiliares basadas en estas, las cuales estructurarán y recuperarán los datos de manera que resulten útiles para el ámbito de la aplicación, dando lugar a **nuestros conectores personalizados.**

### Requerimientos comunes

Para el uso de los conectores (drivers) será necesario tener instaladas las siguientes librerias en nuestro espacio de trabajo:

* Python 3.5
* Librerías de estructuración de datos: *numpy*, *pandas, …*
* Para la representación gráfica: *matplotlib*, *seaborn, tqdm,*

### Comparativa

|  |  |  |
| --- | --- | --- |
|  | Ventajas | Desventajas |
| Cassandra | Permite balanceo de carga entre los nodos disponibles.  Pool de threads para conexiones.  Permite añadir una capa de seguridad.  Conexión única con acceso al clúster de nodos en su totalidad. | <a rellenar> |
| Mongodb | Capacidad de manejo de datos en formatos de documentos, con capacidad dinámica.  Capacidad de transformación de tipos de datos, sobre todo fechas.  Permite añadir una capa de seguridad.  Añade geo espacialidad para los datos relacionados con las coordenadas. | <a rellenar> |
| Neo4j | Permite cambios de configuración de usuario desde el conector.  Pool de conexiones que se sirven a las sesiones.  Posee comunicación mediante TLS y certificados.  Transacciones con simples o auto-commit, y explícitas, con capacidad de auto-reintento. | Driver inmutable, necesidad de crearlo de nuevo cuando hay cambios en la conexión. |

# Análisis y procesamiento de datos

## Metodología

Una vez localizada y decidida de cúal versión de la fuente de datos partiremos, se procede a su **descarga**, **preprocesamiento** o limpieza de datos, para ellao añadiremos fundamentos necesarios para realizar todo el proceso para cada tipo de representación o modelador de datos: Cassandra, Neo4j y/o Mongodb.

* **Análisis de la fuente**. Elección de la versión de la fuente de datos, siguiendo los criterios:
  + Por separación de columnas: .tsv para Cassandra, y .csv para Neo4j y Mongodb
  + Sin filtros añadidos.
* **Descarga**. Por línea de comandos o directa desde el navegador.
* **Preprocesamiento** o limpieza de datos. Mediante *bash scripting* (Cassandra) o mediante scripts de *python*(Mongodb) y *R* (Neo4j).
* **Importación al modelo de datos**. Dependerá del motor que encapsule el modelo de datos en cada caso: Neo4j, Cassandra, Y Mongodb.

## Origen y fuente de datos

Análisis de la fuente y descarga**.** Toda la información que se muestra en el mapa de incidencias, si desactivamos los filtros por defecto, se puede descargar mediante línea de comandos gracias al portal **Socrata**, lo haremos de la siguiente manera:

wget -O incidents.raw.tsv \

"https://data.sfgov.org/api/views/tmnf-yvry/rows.tsv?accessType=DOWNLOAD&api\_foundry=true"

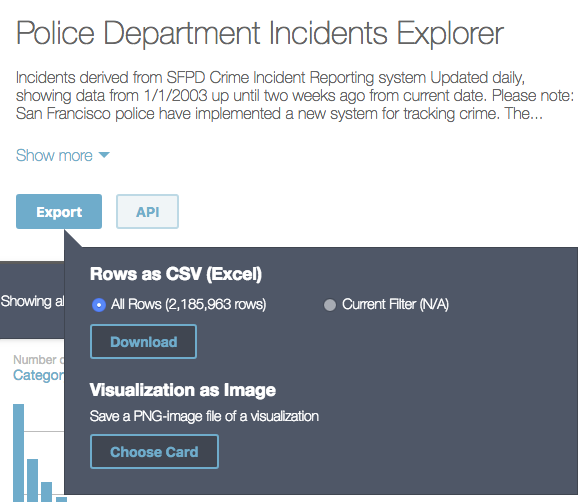
Descarga**.** La fuente de datos necesaria para realizar nuestro análisis de datos se puede descargar desde una página proporcionada por el sistema de datos del departamento policial, a través de la vista ofrecida por el portal de datos **Socrata** **Open Data**, el cual proporciona documentación para extraer reportes y las incidencias mencionadas por descargas manuales o servicios de actualizaciones a través de sus APIs. El informe sobre dicha entrada se encuentra en el portal.

Desde la página que contiene la visualización de un mapa de incidencias basada en este dataset, solamente se pueden descargar versiones del año actual. Esto ocurre desde la fecha del 3 de Marzo del 2018, bajo indicación de la última notificación, ofrecida desde la vista Socrata para dicho reporte:

[Change Notice 03/13/2018]: [Change Notice 03/13/2018]: By the end of this month, this dataset will become historical and a new one will be created starting with incident data in 2018. This one will remain here, but no longer be updated. The new one will have data coming from a new system, will not have a 2 week lag, and have updated districts among other quality improvements. We will attach a guide here with more detailed change updates as soon as we have them.

### ¿Dónde podemos encontrar la versión adecuada?

El dataset, sin filtros y con información desde el 2003 hasta la actualidad (dos semanas antes de la fecha actual), se puede descargar como documento CSV/TSV a partir de una entrada al portal proporcionado por *Socrata*:



Preprocesamiento e importación al modelo de datos

La **limpieza de datos** se ha hecho de acuerdo al modelo de datos requerido para cada motor de modelado de datos: Cassandra, Neo4j o Mongodb; pero en general se ha seguido el mismo patrón de limpieza y han recuperado la mayoría de las columnas.

Con respecto a **la importación de datos**, para cada tecnología se ha seguido una forma de trabajar distinta en acorde al criterio seleccionado y a la discretación de ciertos atributos en acorde al modelo final.

En la fase de **diseño de la estructura** de datos, se procede con la definición del modelo de datos una vez conocidas las sentencias necesarias en la representación y visualización de la informatión, algo que puede ser característico para cada tipo de tecnológia pero a su vez deben de ceñirse a una estrategia común: **visualización de actividad criminal por intervalo de tiempo, distrito y por tipo de delito**, cuyo procedimiento lo podemos encontrar en el apartado de *"Resultados: Representación de la información"*.

A continuación se dispone de las fases y etapas realizadas por cada tecnología para llevar a cabo el procesamiento e importanción y construir finalmente un modelo de datos.

I. Cassandra

II. Mongodb

III. Neo4j

IV. Modelo programático (Cpp)

### I. Cassandra

#### Preparación de la información y limpieza de datos

#### Importación

#### Estructura de datos

II. Mongodb

#### Preparación de la información y limpieza de datos

#### Importación

#### Estructura de datos

III. Neo4j

#### Preparación de la información y limpieza de datos

#### Importación

#### Estructura de datos

IV. Modelo programático (Cpp)

#### Importación

#### Estructura de datos