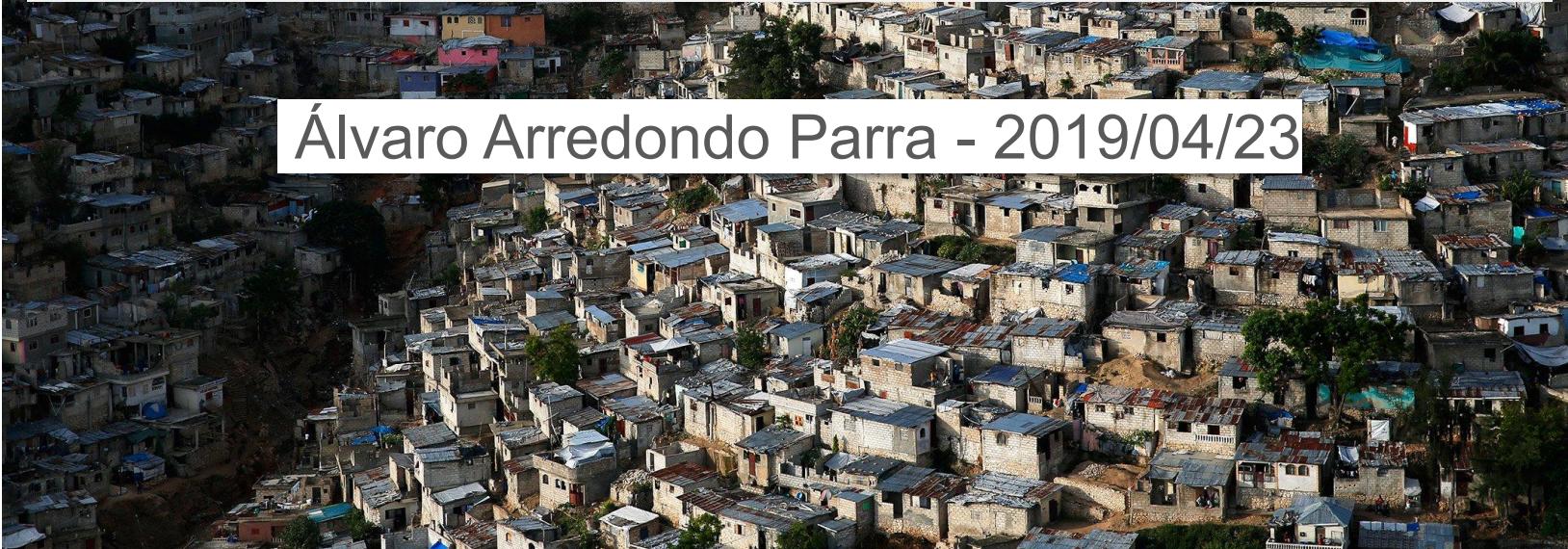


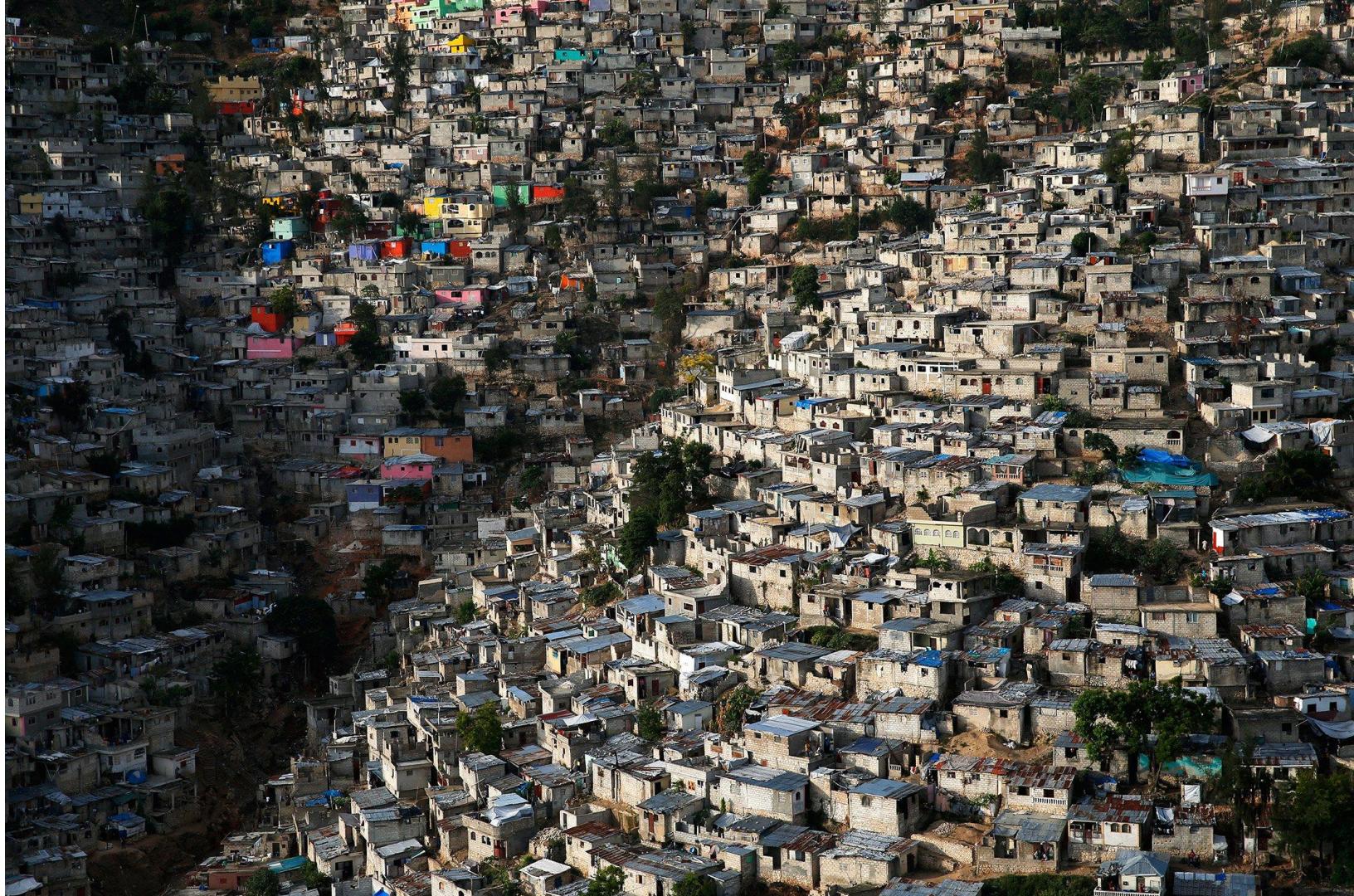


# ¿Y qué casita es?

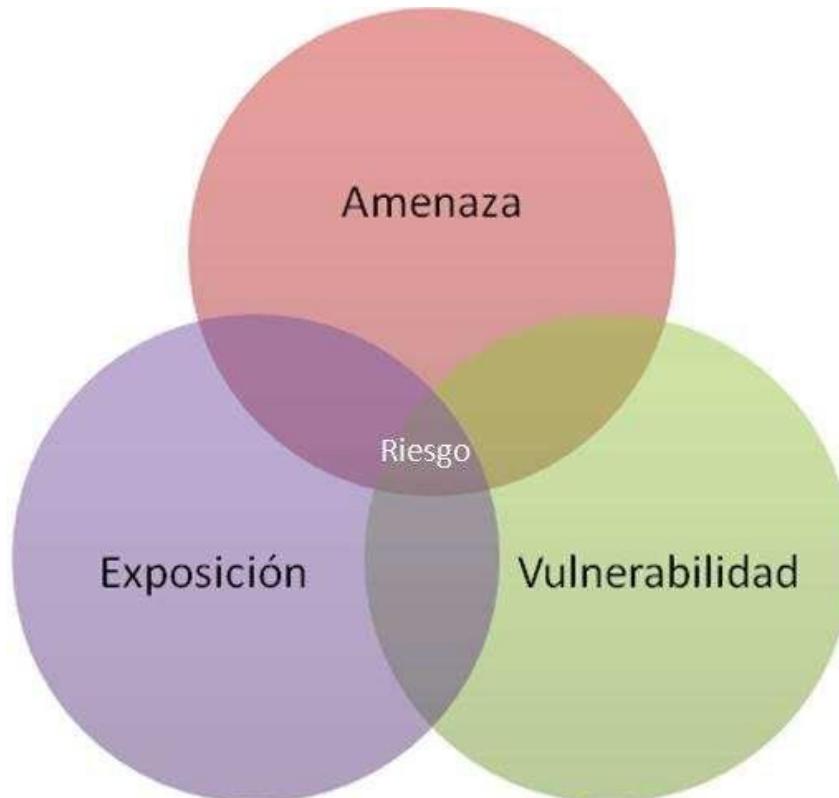
Machine Learning aplicado a imágenes aéreas y LIDAR



Álvaro Arredondo Parra - 2019/04/23



# Vamos a hablar de riesgo



# ¿Por qué?

- En enero 2010, un terremoto de magnitud 7,0 ocurre a kilómetros de Puerto Príncipe
- +300k muertos, +300k heridos  
+1,5M desplazados
- Esfuerzo humanitario inmenso y comienzo del Humanitarian OpenStreetMap



# Necesidades

- Cartografía (HOT OSM)
- Estimar exposición y vulnerabilidad:
  - **¿Cuántos edificios hay?**
  - **¿Dónde están?**
  - **¿Cómo son?**



Abordadas tradicionalmente por la **Ingeniería Civil** a través de **muestreos de campo**.

Mandar ingenieros es **muy costoso**  
Estudios **post-evento**

# Herramientas

- Imágenes aéreas/satélite de alta resolución
  - Fotografías aéreas ortorectificadas, 15 cm/píxel
  - 4 bandas: RGB + Infrarrojo medio (SWIR)
- LiDAR (¡a veces!)
  - 3.4 puntos por m<sup>2</sup>
  - Elevación e intensidad
- Ground Truth: altura, área y tejado
- Python! rasterio, skimage, sklearn, las2py...

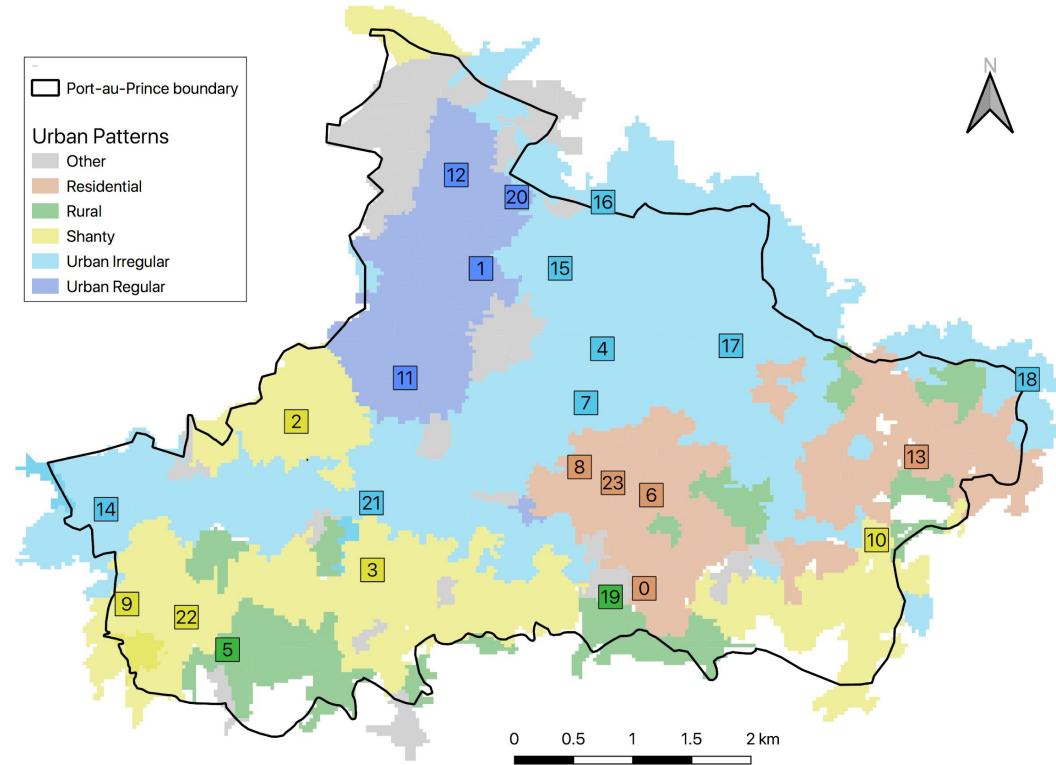


# Herramientas

- **rasterio\***: wrapper (¡simple!) de GDAL para leer rásters como arrays de NumPy
- **pandas**: manipulación de DataFrames (wrapper de NumPy)
- **geoPandas**: wrapper de Pandas con capacidades geoespaciales (depende de **Shapely\*** (geometrías) y **Fiona\*** (sistemas de referencia))
- **scikit-image**: manipulación de imágenes
- **scikit-learn**: modelado (machine learning)
- **laspy**: lectura de nubes de puntos (.LAS/.LAZ)

# Objetivo

- Predecir el número de edificios **por estrato poblacional**
- Predecir la composición de **tipologías constructivas**
- En esta charla nos centraremos en **clasificar los tejados**



# Clasificación por píxel (tradicional)

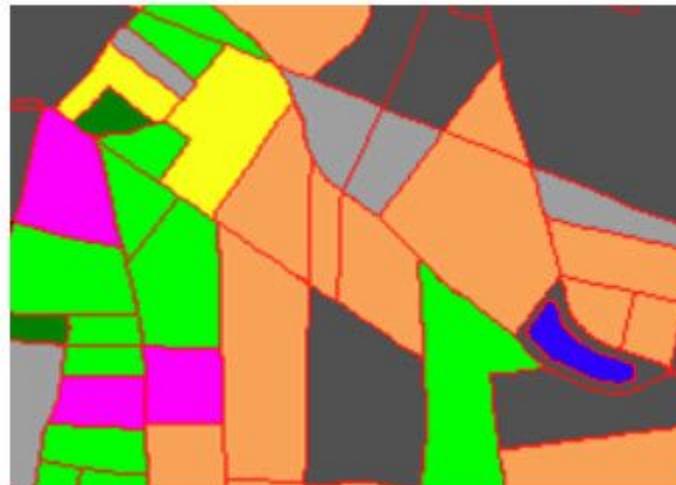
- Cada píxel por separado
- Muy usado en teledetección
- Zonas no delimitadas claramente
- Un atributo por banda



Aplin & Smith 2008

# Clasificación de objetos (Object Based Image Analysis)

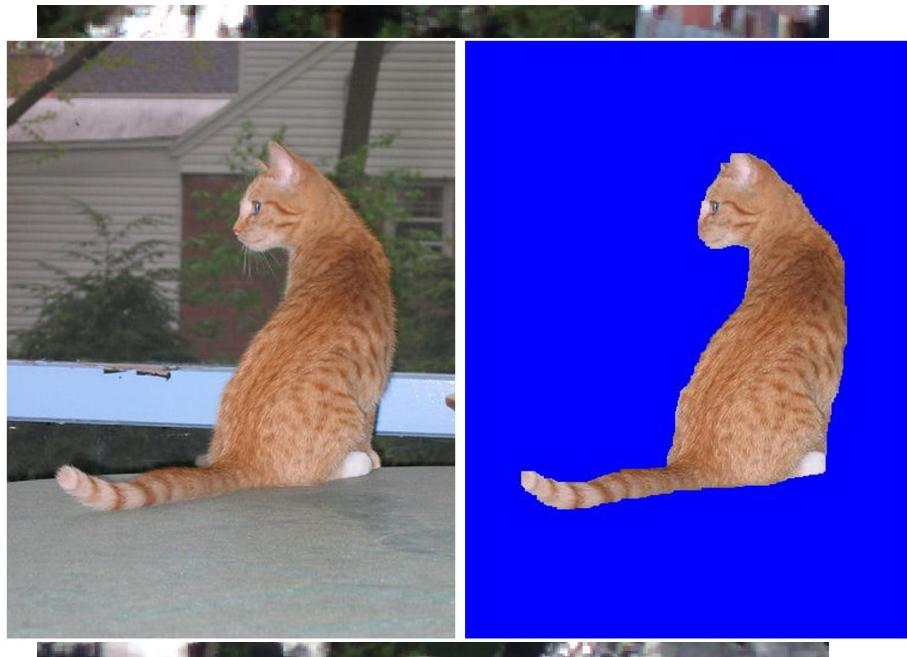
- Requiere **segmentación**
- Objetos claramente delimitados
- Muchos atributos por objeto



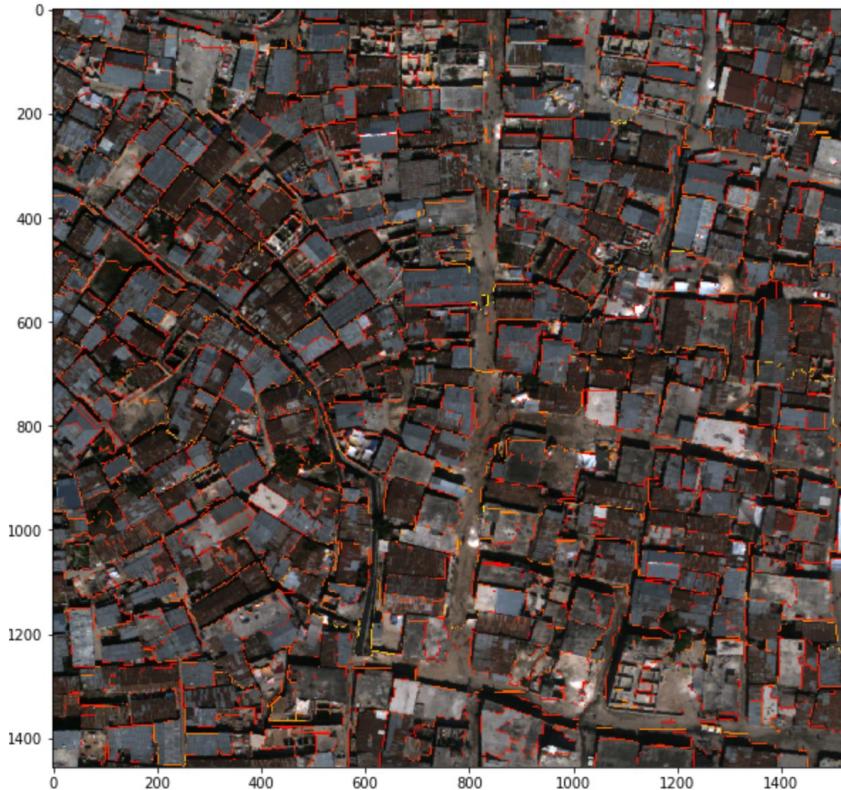
# OBIA: adquirir y preparar datos

- LiDAR
  - Corregir clasificación\*
  - Crear TIN de suelo (malla de triángulos)\*
  - Proyectar alturas\*
  - Convertir a vectorial - **laspy**
  - Convertir a ráster - **gdal\_grid**
- Imágenes:
  - Descargar imágenes: Center for Imaging Science, Rochester Institute of Technology (New York, USA) via FTP - **urlretrieve**
  - Corregistrar imágenes - QGIS
  - Leer imágenes - **rasterio**
  - Quitar bordes (artefactos en SWIR) - **numpy + rasterio**
  - Reescalar imágenes (histogram stretching) - **skimage**
  - Recortar a las muestras - **rio (rasterio)**
  - Alinear bandas (RGB + SWIR + elevación + intensidad) - R!

# OBIA: Segmentación (I)



# OBIA: Segmentación (II)



Claves:

- Introducir el LiDAR en la segmentación

Requisitos del algoritmo:

- Multibanda (6)
- Objetos de diferente tamaño
- Rápido!

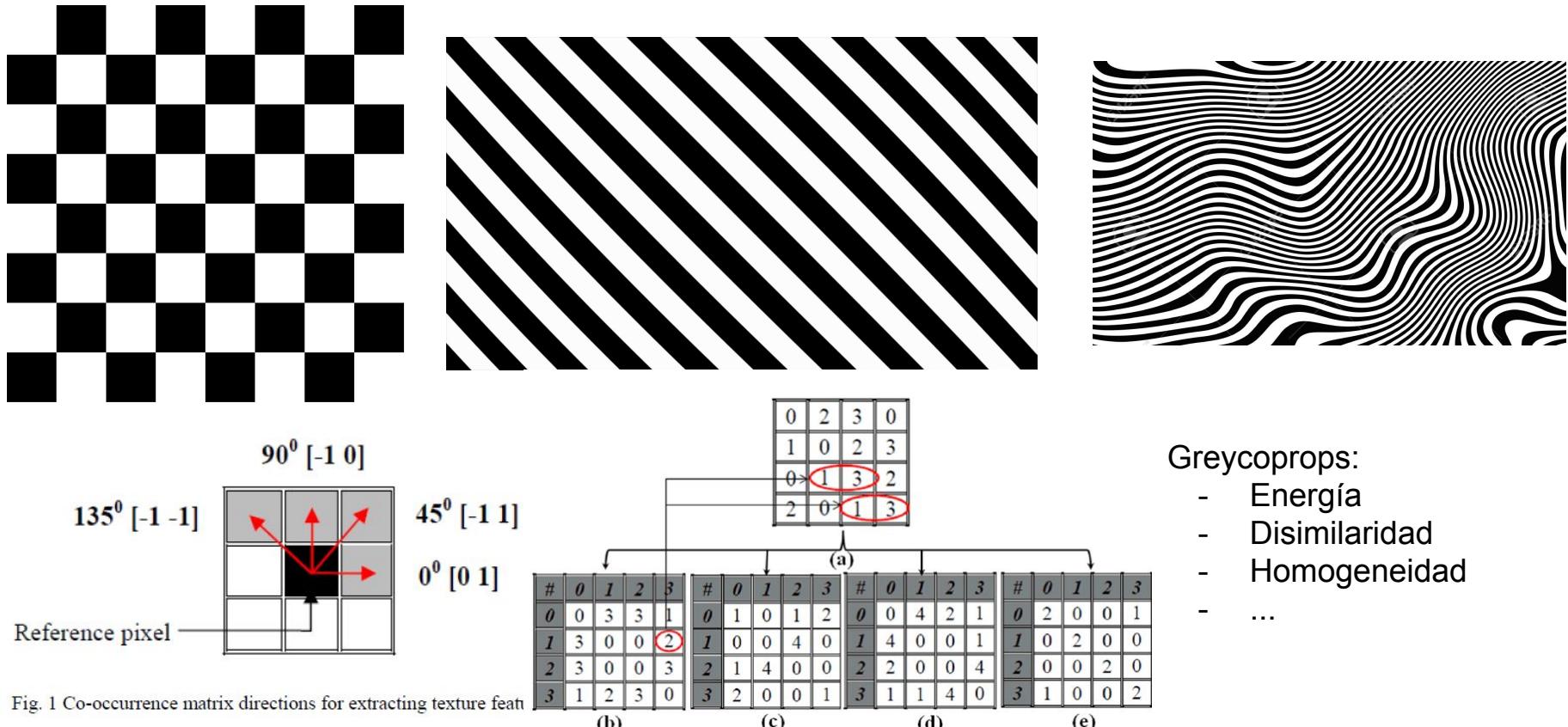
Ganador: **Felzenszwab**

# OBIA: Feature Extraction

Propiedades a extraer de nuestros objetos:

- Forma (por objeto): área, perímetro, coeficiente de forma...
- Radiométricas (por banda): media, mediana, moda, máximo, mínimo, desviación típica...
- Textura (por banda): propiedades de la **matriz de co-ocurrencia de niveles de gris (GLCM)**

# ¿Qué \*\$%&! es la GLCM?



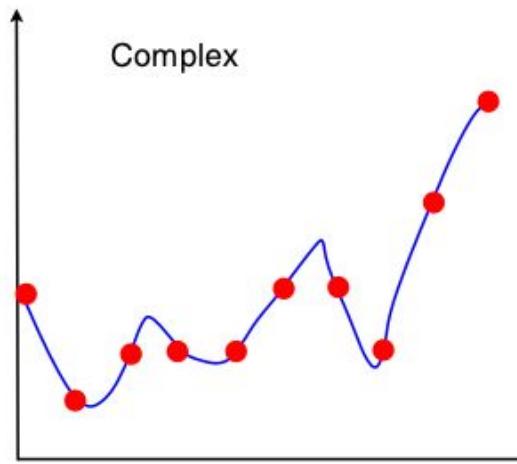
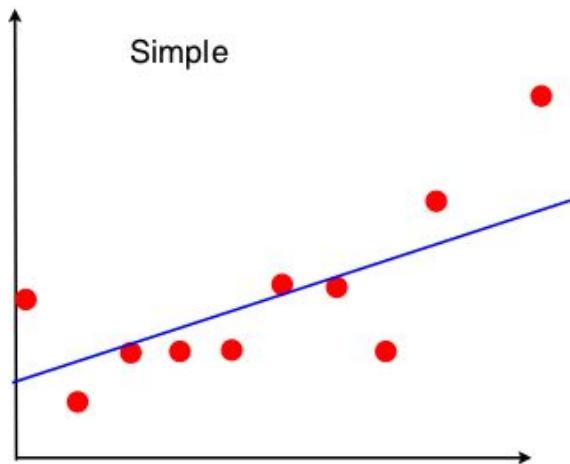
# OBIA: Clasificación (I)

- Clasificación **supervisada**: se enseña al modelo cuál es “la verdad” para que aprenda de ella.
- Puntos de control:
  - Chapa azul
  - Chapa roja
  - Hormigón
  - Suelo
  - Vegetación
  - Sombra



# OBIA: Clasificación (II)

- Modelo sencillo: Random Forest (basado en **árbol de decisión**)
- ¿Por qué?
  - Rápido. Podemos probar decenas de combinaciones de **hiperparámetros**
  - Resiliente contra el **sobreajuste**



# OBIA: Clasificación (III)

```
pipeline = make_pipeline(preprocessing.StandardScaler(),
                        RandomForestClassifier(n_estimators=100))
hyperparameters = {
    'randomforestclassifier__max_depth': [None, 5, 3, 1],
    'randomforestclassifier__max_features': ['auto', 'sqrt', 'log2'],
    'randomforestclassifier__min_samples_leaf': [1, 5, 20, 100]
}
clf = GridSearchCV(pipeline, hyperparameters, cv=10)

# Fit and tune model
clf.fit(x_train, y_train)

print(clf.best_params_)
print(clf.refit)

y_pred = clf.predict(x_test)
print(f'Cohen-Kappa score: {cohen_kappa_score(y_test, y_pred)}\n')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

# OBIA: Clasificación (IV)

Cohen-Kappa score: 0.8975694444444444

	precision	recall	f1-score	support
concrete	0.98	0.89	0.93	47
pavement	0.91	0.94	0.93	34
shadow	0.83	0.86	0.84	22
tin	0.91	0.98	0.95	53
vegetation	0.95	0.86	0.90	21
micro avg	0.92	0.92	0.92	177
macro avg	0.92	0.91	0.91	177
weighted avg	0.92	0.92	0.92	177

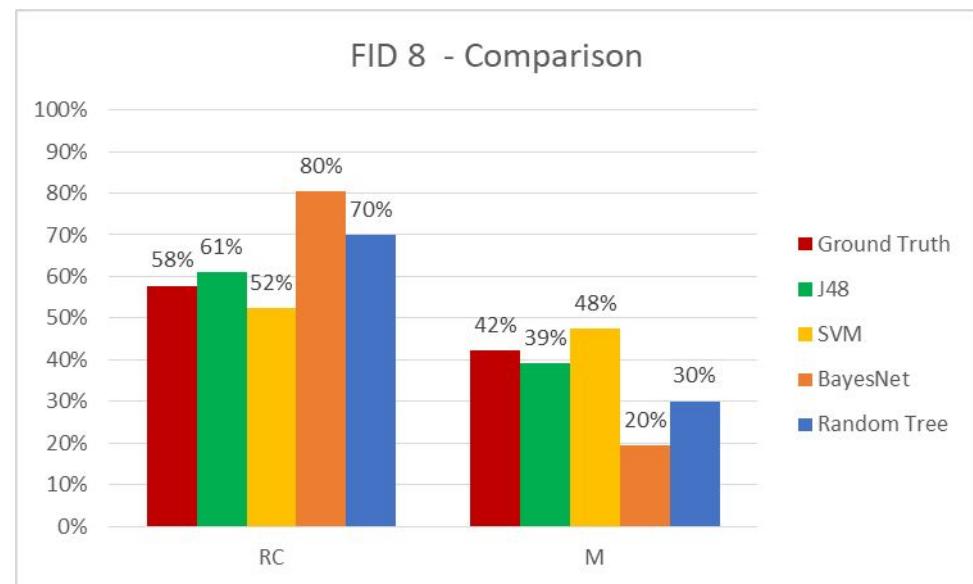
# Ya tenemos los objetos... ¿ahora qué?

- Hemos extraído las variables necesarias (**altura, área y material del tejado**) para construir un modelo que prediga la **tipología constructiva**
- En Haití hay principalmente dos tipologías:
  - Estructura de **hormigón armado**
  - Estructura de **mampostería reforzada con madera**
- Solo tenemos tres atributos: podemos usar un árbol de decisión. Debería ser fácil, ¿no?

:()

- No pasamos del 70-80% de F1 Score

- ¿Qué está pasando?





Intentamos modelar la realidad  
pero **la realidad es caótica**



# Conclusiones

- Introducir datos LiDAR en la segmentación (y no solo en la clasificación) la **mejora brutalmente**
- Conseguimos identificar edificios con **gran precisión (> 95%)**
- Los resultados negativos también dan información y nos ayudan a **centrar nuestros esfuerzos** en las zonas más complejas
- Python tiene **multitud de librerías** para manejar datos geoespaciales, analizarlos y extraer resultados