ANALISIS DE DATOS SISMOGRÁFICOS PROCEDENTES DE LA MISIÓN VIKING

INTRODUCCIÓN

Lo primero que haremos es realizar un estudio de los datos recopilados por el proyecto Viking durante su vida útil en la superficie de Marte.

Una vez hecho el estudio, se realizará un clasificador capaz de analizar a tiempo real un flujo de datos con tal de valorarlos.

Esto surge debido a que no se tenía en consideración las rachas de viento encontradas en el planeta.

El brazo en el que se encontraba el sismógrafo, estaba unido al caparazón del módulo, por ello que, al vibrar, daba lecturas erróneas al sismógrafo.

Por ello que queramos clasificar qué lecturas son válidas de las que no.

INDICE

1. Fase de análisis
   1. Procedencia de los datos
   2. Estructuración de los datos y estadísticas
      1. Estructura de los archivos
      2. Estructura de los datos
      3. Lectura del dataset
      4. Estadísticas básicas de variables a analizar
   3. Agrupación de datos
      1. Normalización del dataset
      2. Matriz de correlación
      3. Análisis de relaciones lineales
      4. Agrupación de variables como grupos independientes
   4. Variables no relacionadas linealmente
2. Fase de modelado de algoritmos no supervisados
   1. Reducción de variables dependientes a independientes
      1. Introducción PCA
      2. Aplicación de PCA sobre cada grupo de variables dependientes
      3. Validación de los datos
   2. Modelos de clasificación no supervisados.
      1. Introducción modelos no supervisados
      2. Modelo K-Means
         1. Introducción modelo K-Means
         2. Determinación de K
         3. Modelado
      3. Modelo GMM
         1. Introducción modelo GMM
         2. Modelado
3. Fase de predictor y generador de datos para simulación
   1. Introducción Keras
   2. Montaje de generador de datos para simulaciones en flujo
   3. Creación del servidor
4. Clasificación en flujo
   1. Puesta a punto del servicio de clasificación cliente – servidor
   2. Funcionamiento general del sistema

1. FASE DE ANÁLISIS

1.1 Procedencia de los datos:

El programa Viking fue una de las misiones más ambiciosas lograda por EEUU.

Dicho programa constaba de dos sondas, cada una de ellas formada por un orbitador y un módulo de aterrizaje. Ambas eran exactamente iguales, por ello que se las denominaran sondas gemelas.

Para tratar sobre ellas, siempre que se muestren datos en parejas, la primera se referirá a Viking 1 y la siguiente a Viking 2.

Los aterrizadores lograron aterrizar en lugares diferentes, una lo logró en agosto de 1975 y la siguiente en septiembre del año 1975 también.

Su misión principal era lograr fotografías del terreno, obtener datos básicos que sirvieran para recopilar información y un conjunto de 3 experimentos biológicos.

Todo estaba planeado para lograr recopilar datos suficientes a lo largo de 90 días tras el aterrizaje.

Al final, los orbitadores lograron transmitir datos hacia el planeta Tierra , una hasta el año 1980 y la otra hasta el 1978.

Referente a los módulos de aterrizaje, la Viking 1 retransmitió datos a la Tierra hasta el año 1980, y la Viking 2 hasta el 1982.

1.2 Estructuración de los datos y estadísticas:

Todos los datos recopilados del proyecto Viking están en un servidor público perteneciente a la universidad de Washington y dados por la NASA.

El enlace que usaremos para descargar los archivos es:

<https://pds-geosciences.wustl.edu/missions/vlander/seismic.html>.

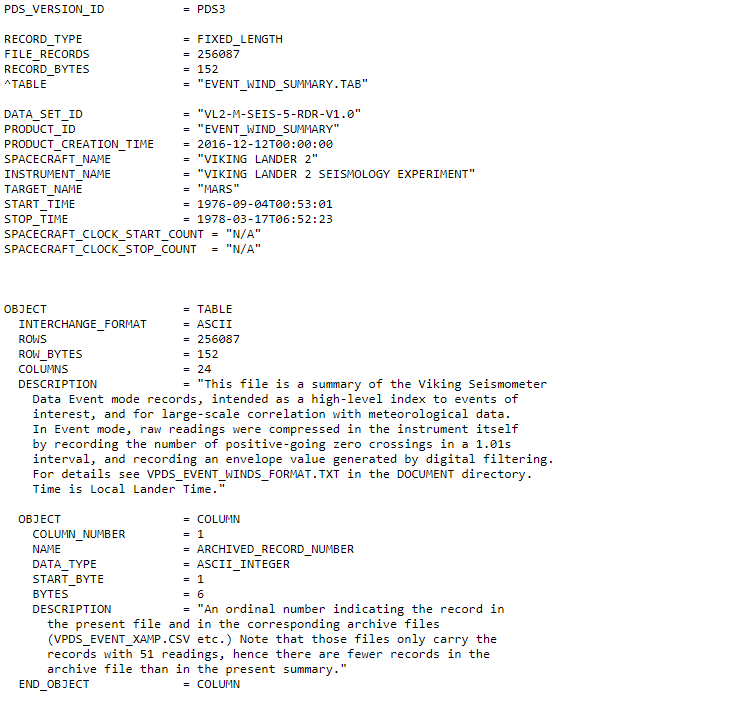
1.2.1 Estructura de los archivos

Los archivos podremos encontrarlos en tres formatos diferentes: csv, lbl o tab.

Los archivos csv son archivos comúnmente utilizados, separados los datos por columnas y con un separador común. En nuestro caso es la coma.

Los archivos tab, es el otro tipo contenedor de datos, donde cada dato está separado por un número de bytes establecidos en el archivo lbl y éste cambiará según la columna y su contenido.

El tercer tipo de archivo lbl, contendrá información sobre el documento al que referencia, que datos tiene de cada columna, en que byte empieza y en cual acaba, y el tipo de datos que contiene.



1.2.2 Estructura de los datos

Los que nos interesarán en concreto será el high\_wind\_summary y el event\_wind\_summary.

En ellos podremos encontrar las siguientes variables, tal y como describe su archivo lbl correspondiente:

Variables temporales:

1. SEISMIC\_TIME\_SOLS: es una variable que engloba la escala de tiempo en soles decimales para datos sismográficos.

Su ecuación es: sol+(hr\*3600.0+min\*60.0+sec) /88775.0

1. METEO\_TIME\_SOLS: es una variable que engloba la escala de tiempo en soles decimales para datos meteorológicos.

Su ecuación es: sol+(hr\*3600.0+min\*60.0+sec) /88775.0

1. DATA\_ACQUISITION\_SOL: indica en que día marciano (sol) fueron adquiridos los datos, tomando como el sol 0 el día de aterrizaje.
2. DATA\_ACQUISITION\_HOUR
3. DATA\_ACQUISITION\_MINUTE
4. DATA\_ACQUISITION\_SECOND
5. WIND\_SEISMIC\_INTERVAL: tiempo en segundos transcurridos entre adquisición de datos de lectura del viento y datos sismográficos. Tomar unicamente cuando este valor sea positivo.

Variables meteorológicas:

1. WINDSPEED: velocidad del viento en m/s.
2. PRESSURE: presión atmosférica en mbar.
3. WIND\_DIRECTION: dirección del viento relativa al viento en grados.
4. AIR\_TEMPERATURE: temperatura del aire en kelvin.

Variables sismográficas:

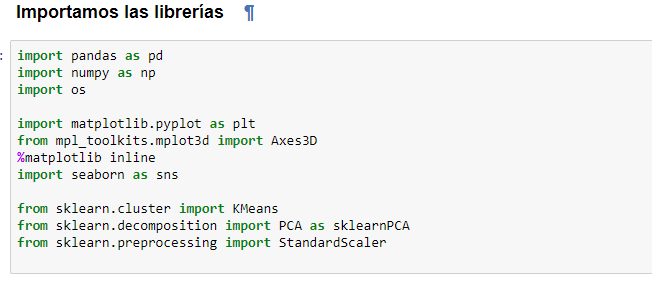
1. FIRST\_X\_AXIS: primera lectura tomada del sismografo en el eje X.
2. MEDIAN\_X\_AXIS: La media de valores tomados en el eje X. Cada valor esta medido en digital unit (DU) y se corresponde a 2 nm tomados a 3Hz.
3. MAXIMUM\_X\_AXIS, MINIMUM\_X\_AXIS: valor máximo y mínimo de las lecturas tomadas. Pueden ser tanto valores positivos o negativos, en el eje X.
4. RMS\_X\_AXIS\_X100: valor eficaz o valor cuadrático medio. Nos permite calcular la magnitud de unos valores discretos en valores positivos.
5. MEAN\_X\_AXIS\_CROSSINGS: La media de valores en los que la onda toma el valor 0 en el eje descrito. En este caso será la variable X.

1.2.3 Lectura del dataset:

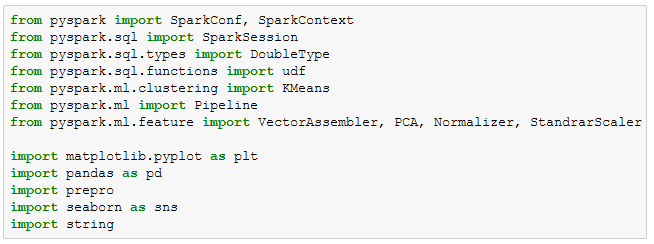
Para empezar a analizar los datos, deberemos de leer dichos datos del dataset seleccionado. Empezaremos haciendo uso del archivo EVENT\_WIND\_SUMMARY.

Debido al formato declarado anteriormente de los archivos tab y lbl, en Python no se pueden leer directamente por lo que hemos juntado ambos en un archivo csv.

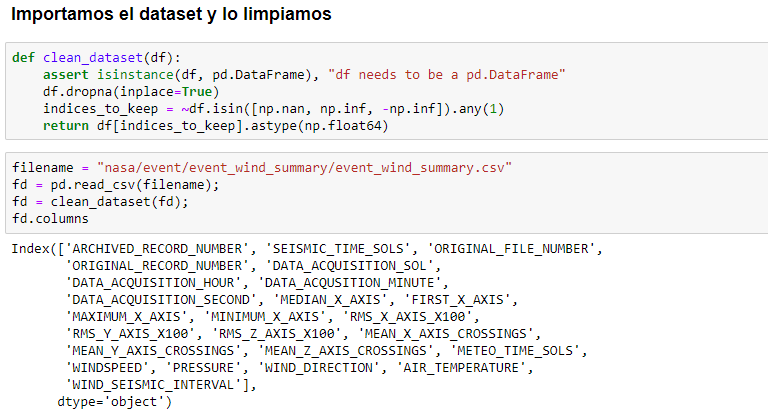
Lo primero será importar todas las librerías necesarias para nuestro proyecto:



Para la versión en **Spark,** las libreiras necesarias son las siguientes



Una vez importadas, leemos los datos, limpiamos los nulos, vacíos, etc en el caso en el que existan y sacamos la cabecera:

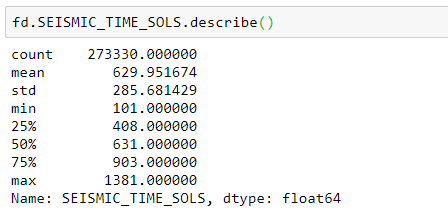


Sacamos las estadísticas básicas haciendo el uso de describe para cada variable como se indicará en el siguiente apartado.

1.2.4. Estadísticas básicas de las variables a analizar

*Sobre las variables temporales:*

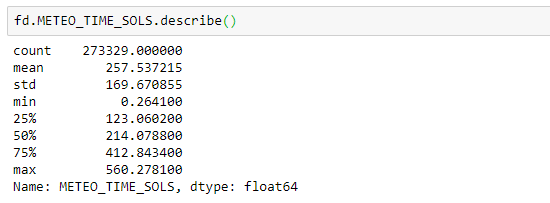
1. SEISMIC\_TIME\_SOLS:



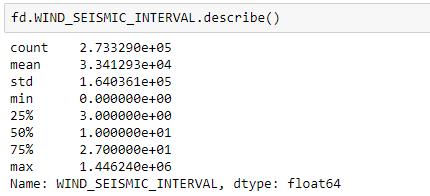
Podemos ver que los valores mínimos están en 101 y los máximos en valores 1381 soles decimales.

Observando los percentiles podemos decir que los valores están distribuidos por la normal de forma uniforme.

1. METEO\_TIME\_SOLS:



1. WIND\_SEISMIC\_INTERVAL:



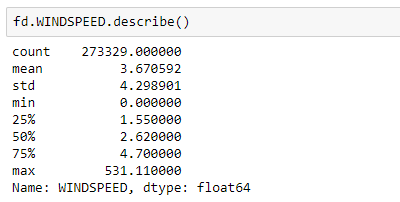
En la documentación se expone que los valores en las ultimas tomas y en las primeras tomas del dataset, no tenía que tenerse en cuenta debido a que los valores eran excesivos.

Esto hace que la media y la desviación estándar inicial no pueda usarse.

Tomaremos como valores entre 0 a unos 27 segundos.

*Sobre las variables meteorológicas:*

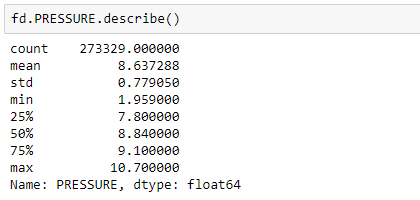
1. WINDSPEED: velocidad del viento en m/s.



La velocidad del viento varía entre 0 m/s hasta los 531 m/s. Este valor no tiene sentido. Por lo que seguramente nos tocará hacer una limpieza inicial de valores.

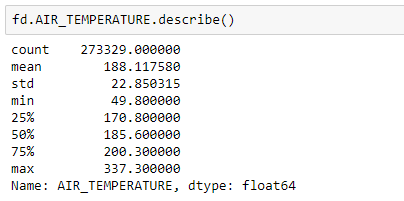
Viendo los percentiles, hasta un 75% de los datos tomados están por debajo de los 5m/s, valores con vientos de valor bajos – medios.

1. PRESSURE: presión atmosférica en mbar.



La presión irá desde los 0 milibares hasta los 10,7.

1. WIND\_DIRECTION: dirección del viento relativa al viento en grados que irá de 0 a 360º.
2. AIR\_TEMPERATURE: temperatura del aire en kelvin.

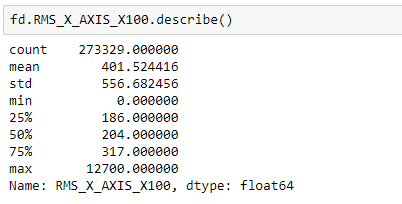


Por lo que podremos ver, la temperatura del aire irá desde 50 kelvin hasta 337k .

*Sobre las variables sismográficas:*

1. RMS\_X\_AXIS\_X100:

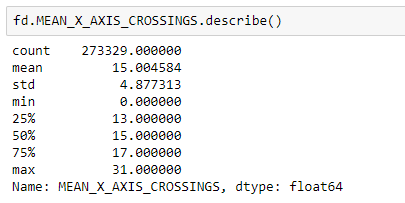
Debido a que los valores van desde unos -130 hasta +130, usaremos el RMS para sacar una magnitud de dichos valores. Sus estadísticas por tanto son:



Podemos observar que van desde 0 DU hasta 12700 DU.

Tal y como muestran los percentiles, tomaremos en cuenta hasta unos 350 DU. Los mayores serán valores sin sentido o no concluyentes.

1. MEAN\_X\_AXIS\_CROSSINGS:



Se puede ver que irá desde 0 hasta los 31. Cuanto más rápido oscile la onda, mayor será el número de pasos por 0.

Este tipo de valores puede usarse para el estudio sonoro de fondo para la detección de sonidos, estructuras o efectos en tiempos discretos.

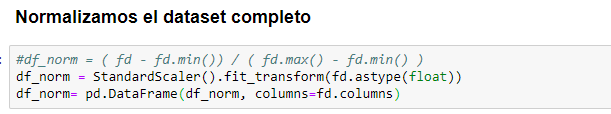
* 1. Agrupación de datos

1.3.1 Normalización del dataset

Nuestro dataset tiene variables que describen efectos diferentes. Cada una tomará una distribución diferente de datos.

Para que todas escalen entre los mismos valores, usaremos la normalización o estandarización.

Para hacerlo en Python:



En la versión de **Spark** lo vamos a hacer más tarde, una vez hayamos formado los grupos con los datos correlacionados, ya que esto facilita el código.

1.3.2 Matriz de correlación

Trataremos de encontrar relaciones lineales entre pares de variables, dejando en dichos pares las demás variables como constantes.

Esto nos permitirá encontrar relaciones y grupos entre variables.

Debido a que en el modelo de clustering todas las variables de entrada tendrán que ser independientes, deberemos de encontrar grupos en primer lugar para quitar la dependencia entre esas variables.

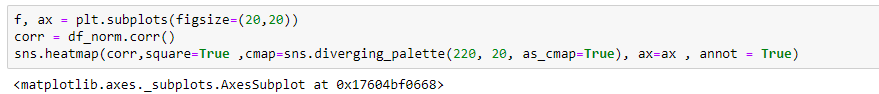
Para todo esto haremos uso de las matrices de correlación.

Cada valor tomará valores decimales entre -1 y 1.

Cuando más se acerque a 0, menos relación lineal habrá entre variables.

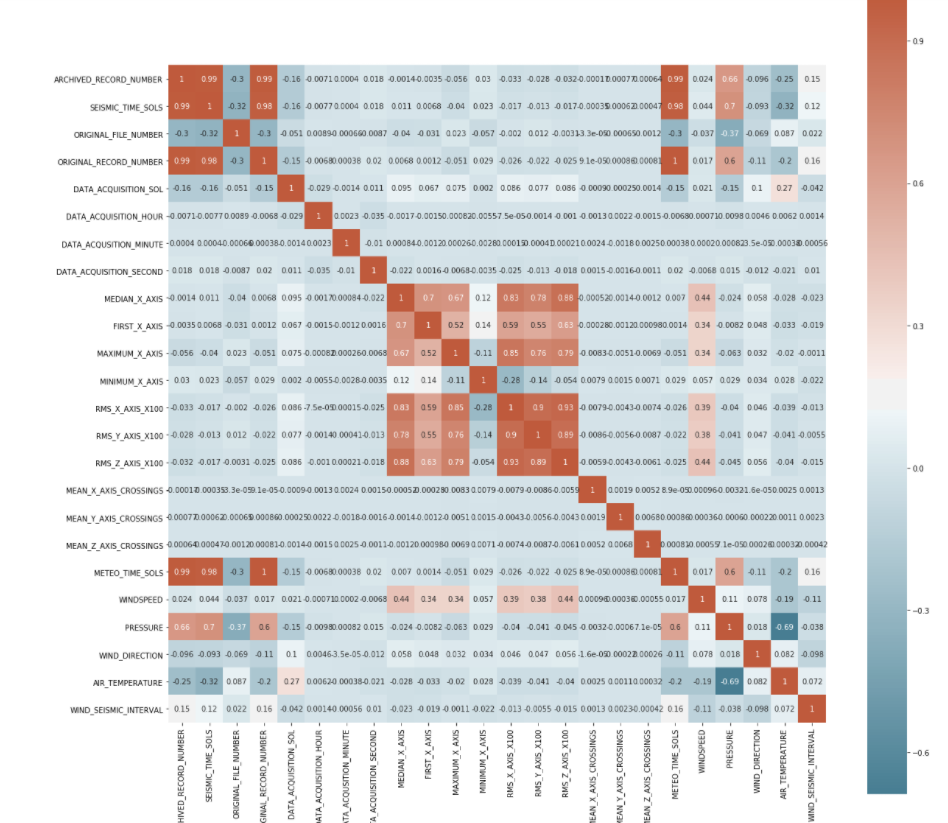
Cuando más se acerque a -1, mas relación inversa habrá entre variables (Al aumentar una, disminuye la otra), y cuando más se acerque al 1, el caso contrario.

Podremos realizarlas en Python con:



1.3.3 Análisis de correlaciones lineales entre variables:

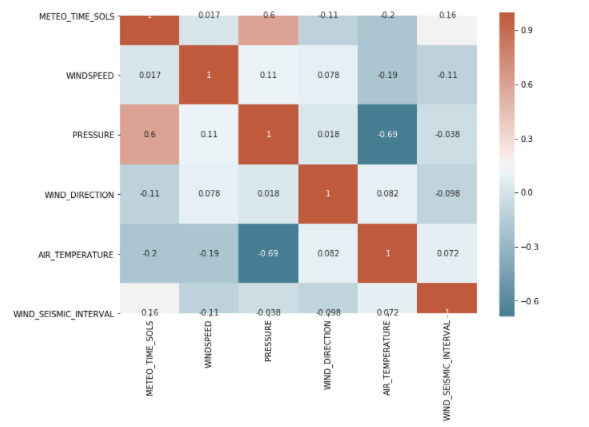
El resultado será el siguiente gráfico:



Es una tabla bastante amplia, pero la usaremos únicamente como visión general a la hora de hacer los grupos.

Tomaremos los valores como correlacionados fuertemente como 1.0.5 y -0.5…-1, una correlación débil entre 0.3 … 0.5 y -0.5 … -0.3 y sin correlación hasta el 0.

*Correlaciones de variables meteorológicas:*



Podemos ver que hay una relación inversa entre la temperatura y la presión en un rango de correlación alto.

También hay una relación directa entre la presión y la variable temporal de adquisición de datos meteorológicos.

Habría que discutir más adelante si introducir las variables temporales en el modelo, ya que las variables al fin y al cabo varían según este, pero no todas de forma lineal como lo hace la presión, por lo que puede sesgar el modelo.

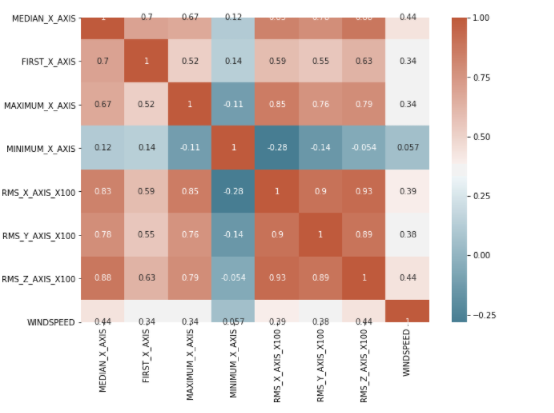
De este conjunto agruparemos únicamente la presión y la temperatura, sin tener

en cuenta la variable temporal.

*Correlaciones de variables temporales:*

Debido a que son escalas temporales, la de segundos, minutos, horas y soles serán independientes entre ellas pero serán dependientes sobre todo de METEO\_TIME\_SOLS y de SEISMIC\_TIME\_SOLS debido a que se crean a partir de las mismas.

*Correlaciones de variables sismográficas:*



Recordemos que el sismógrafo toma lecturas en tres ejes, eje X, eje Y, eje Z.

Se ve muy claramente que todas las variables están relacionadas entre ellas fuertemente, y que el valor del viento también les afecta.

Esto quiere decir que los valores del viento afectan de forma lineal a los valores tomados del sismógrafo.

1.3.4 Agrupación de variables como grupos independientes

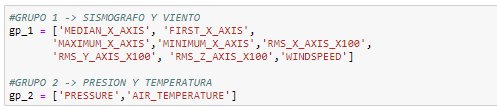
Una vez hecho el estudio de correlaciones, dividiremos el dataset en pequeños subgrupos.

Todas las variables dentro de cada subgrupo estarán dentro de un grado de correlación, relacionada entre ellas.

Cada uno de los subgrupos serán independientes entre ellos.

Como un modelo solo puede tener de entrada variables independientes, aplicaremos PCA a cada subgrupo de variables dependientes para dejarlo en una sola variable.

En Python referenciaremos a dichas variables con:



* 1. Variables no correlacionadas linealmente

Para tratar sobre la atmósfera de Marte y poder analizarla, deberemos de considerar ciertos valores como la temperatura, la presión y la velocidad del viento.

El aire, el cual está formado de materia y gases, que compone la atmósfera ejerce una presión sobre los objetos de la superficie. Cuanto más pese el aire, más presión ejercerá sobre la superficie. A esta presión se la denomina presión atmosférica, y se mide a partir de un barómetro. Sus unidades de medición pueden ser en bares o en pascales entre otras.

Dicha presión atmosférica disminuye al aumento de altitud, la humedad y la temperatura.

Al variar esta presión atmosférica se produce las corrientes de aire. Esto es debido a que al ejercer el aire más peso en un lugar que en otro, hace que exista un movimiento de materia y gases en la atmósfera, produciendo en tanto una corriente.

En Marte, la presión atmosférica es muy baja. Unas 120 veces más baja que la de la Tierra. Por ello que se necesite una mayor velocidad de viento para poder levantar polvo. Aunque en la Tierra se necesite una velocidad de unos 50/60 km/h, en Marte se necesitará unos 150-200 km/h. Esto también afecta en gran modo a las vibraciones sobre el módulo.

Surge una duda entonces de todo, ¿Cómo puede ser que exista velocidades de viento tan altas con una presión tan baja en Marte?

Esto es debido a esas variaciones tan grandes de temperaturas al día, ya que pueden varias de unos 10ºC a -100ºC al día.

Tras todo esto, aunque aparezcan que el viento sea independiente de la presión y la temperatura en el estudio de correlaciones, están relacionadas de una forma no lineal.

Lo mismo ocurre con las variables temporales.

Estas relaciones no lineales no afectarán en principio al entrenamiento del modelo, aunque estarán metidas de forma indirecta.

2. FASE DE MODELADO

2.1 Reducción de variables dependientes a independientes

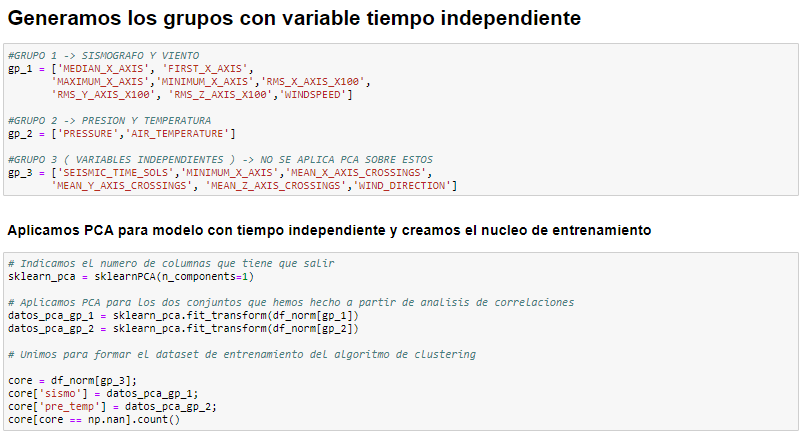
2.1.1 Introducción PCA

La funcionalidad de aplicar PCA o análisis de componentes principales es describir las características de un conjunto de variables y reducirlas a un conjunto de variables no correlacionadas de dimensiones menores.

Debido a que para entrenar un modelo no supervisado deberán ser todas sus variables de entrenamiento independientes, deberemos de aplicar a cada subgrupo hecho anteriormente PCA.

2.1.2 Aplicación PCA sobre cada grupo de variables dependientes

En Python lo lograremos con:

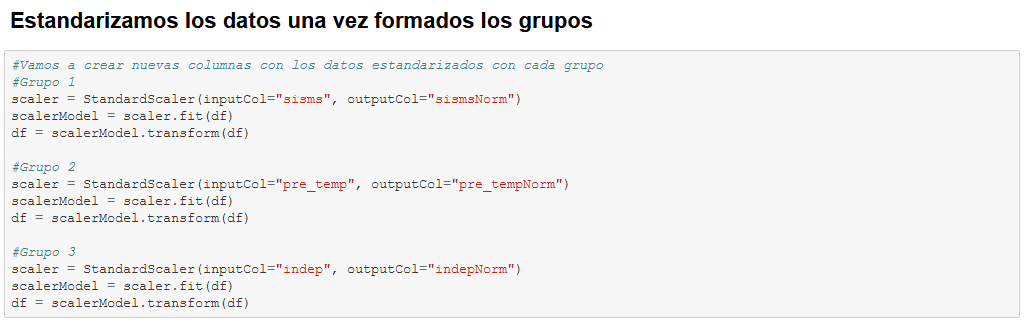


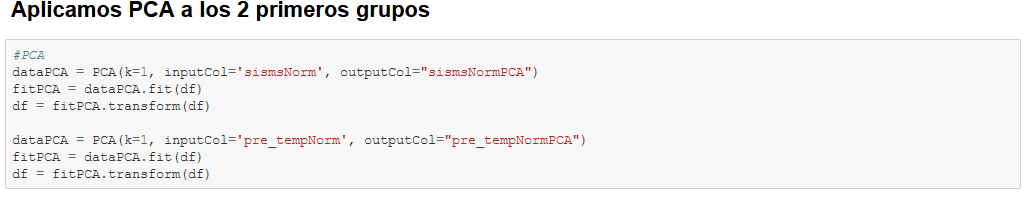
Con n\_components lo que hacemos es reducir a 1 variable el grupo 1 y a otra sola variable el grupo 2. A la primera variable la denominaremos sism, y a la seguntra pre\_temp. En el grupo 3 tendremos todas aquellas variables independientes que queremos mantener.

Una vez aplicado el pca, añadimos a las variables independientes, la variable sismo y pre\_tem.

En la versión de **Spark,** esto mismo lo hacemos de la siguiente manera:



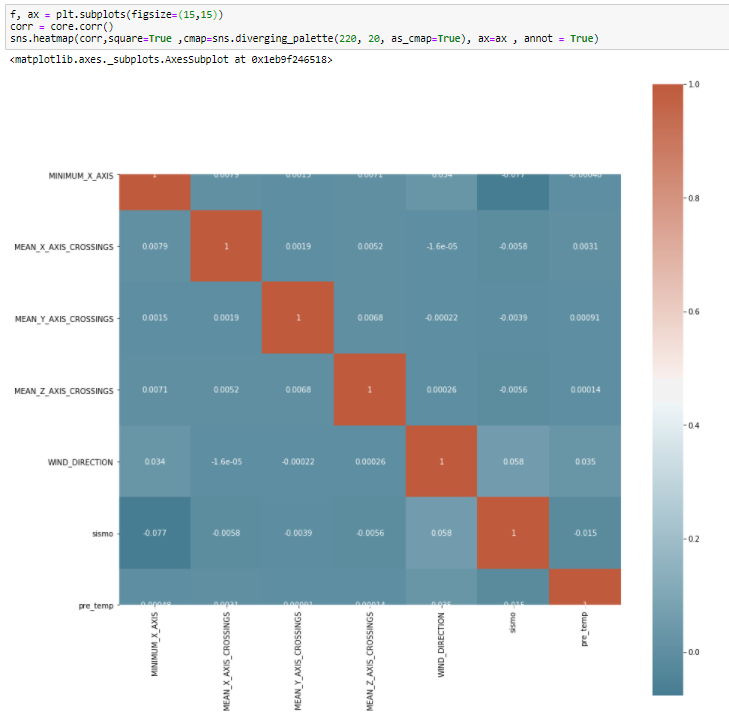




Además se observa que antes de aplicar PCA y una vez formados los grupos, se estandarizan los tres. Esto se hace así en lugar de antes para conservar los datos originales junto con los estandarizados, lo cual nos facilitará el trabajo considerablemente a la hora de representar gráficamente los resultados.

2.1.3 Validación de los datos

Ahora en core podremos comprobar que todas las variables son independientes, estando todas entre +0.1 y -0.1.



2.1 Modelos de clasificación no supervisados

2.2.1 Introducción modelos no supervisados

Los modelos no supervisados permiten buscar patrones entre los datos que tenemos, sin la necesidad de que estos estén etiquetados. Como solo tenemos los datos de entrada y no datos de salida utilizar este tipo de modelo tiene como finalidad describir la estructura de los datos para encontrar algún tipo de organización que simplifique el análisis.

Estos algoritmos también suelen tomar el nombre de algoritmos de clustering, ya que intentan formar grupos (clusters) a partir de los datos con características similares.

2.2.2 Modelo K-Means

2.2.2.1 Introducción del modelo K-Means

K-Means es uno de los algoritmos de clustering más populares. Intenta dividir unos datos en k grupos, siendo k un número seleccionado por el usuario, en el cual cada observación pertenece al grupo con la media más cercana.

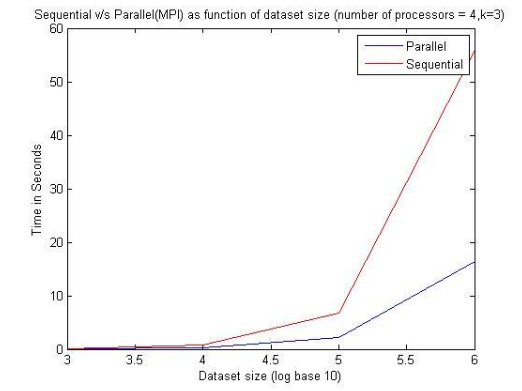
El algoritmo funciona de la siguiente manera:

- Se inicializan los centros de los clusters, por ejemplo, de manera aleatoria.

- Se hacen varias iteraciones de lo siguiente: Se asigna a cada punto los centroides, cada dato al que tenga más cercano. Una vez hecho esto, se actualizan los centroides calculando la posición promedia de todos los elementos en ese grupo.

- Se repite este último paso haya que los centroides apenas se muevan, es decir, se haya alcanzado el resultado óptimo.

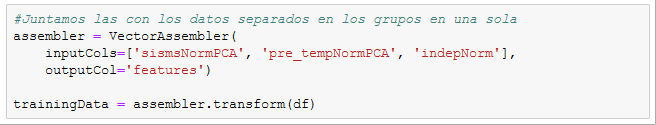
Es un algoritmo muy costoso computacionalmente, con una complejidad de O(n^2). Además, por la forma en la que se implementa, requiere comunicación constante entre nodos, lo cual lo hace difícil de paralelizar. Sin embargo, no es imposible, y Spark implementa en su librería una versión paralelizable de este algoritmo.



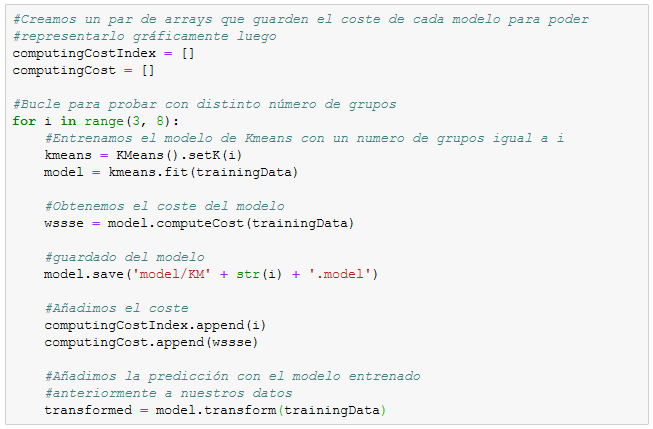
Este gráfico muestra una la mejora experimentada gracias a utilizar una versión paralelizable, en el que se observa que la mejora no es especialmente grande por el problema discutido anteriormente, pero sí bastante significante.

2.2.2.3 Modelado

En **Spark** se hace de la siguiente manera:



Aquí lo único que hacemos es juntar los 3 grupos en una columna llamada “features”. Esto se debe a que la función de la librería de Spark requiere que todas las características estén en una única columna.

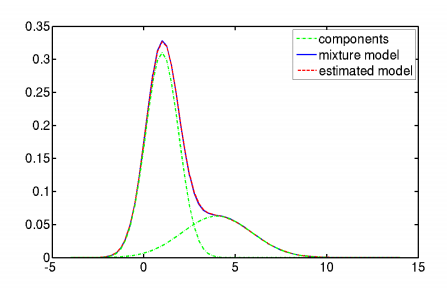


En este fragmento de código observamos cómo no utilizamos sólo un único número de clusters para el kmeans, sino que probamos con varios, para ver cuál produce mejores resultados sin excesivo sobre ajuste.

2.2.3 Modelo GMM

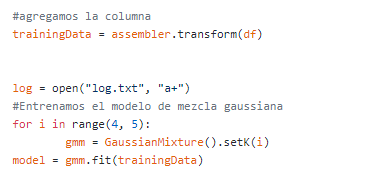
2.2.3.1 Introducción modelo GMM

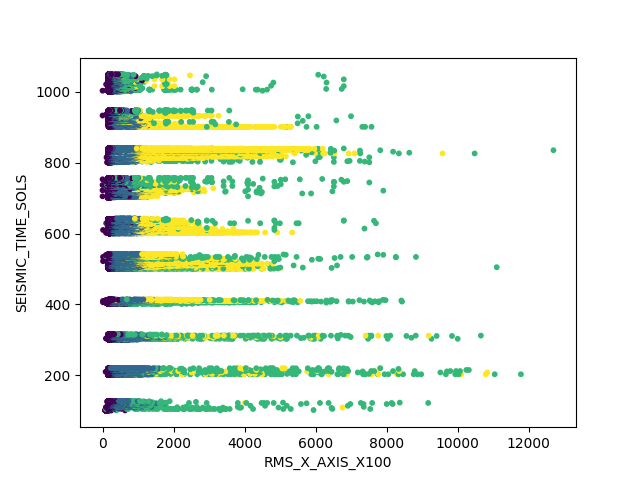
El modelo GMM o modelo de mezcla Gaussiana es un modelo probabilístico en el que todos los puntos de datos se generan a partir de un número finito de distribuciones gaussianas. La finalidad de usar este tipo de modelos es aproximar o estimar a partir de sus componentes encontrando una similitud respecto a los datos que contiene las componentes.



Ejemplo de uso de GMM a partir de dos componentes.

2.2.3.1 Modelado





Salida generada