UNSA

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

Escuela Profesional de Ciencia de la Computación

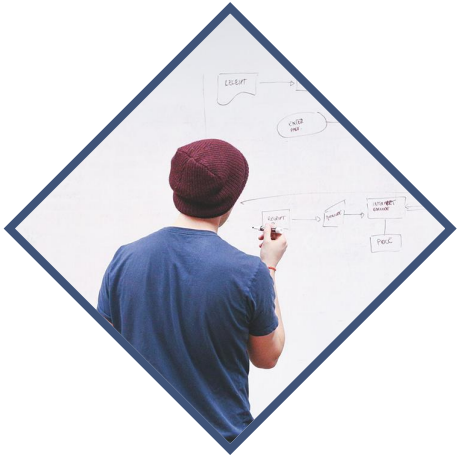
# Proyecto Final : Reconocimiento de actividad humana utilizando datos de acelerómetro de sensores portátiles

Curso : Tópicos en Base de Datos
Docente : PhD. Edgar Sarmiento Calisaya

Alumnos : Miguel Alexander Herrera Cooper
Milagros Celia Cruz Mamani
Yordy Williams Santos Apaza
Alexander Cordova Ccana

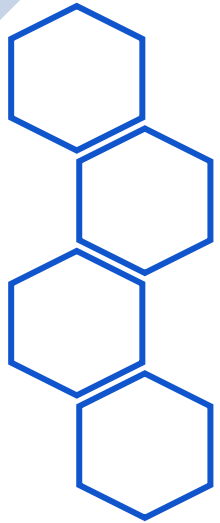
School of Computer Science
UNSA

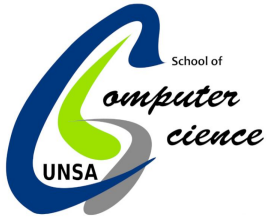
# INDICE



# 1

# Introduccion





# Introduccion : ¿Qué es el reconocimiento de la actividad humana?

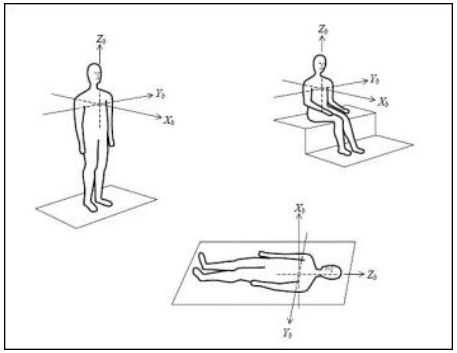
- **Reconocer múltiples conjuntos de actividades humanas diarias en condiciones del mundo real.**

- **¿Qué dispositivos se utilizan para recopilar datos para el reconocimiento de actividades humanas?**
  - Smartphones
  - Wearable devices

- Cada uno de estos dispositivos tiene un acelerómetro integrado (biaxial / triaxial) que realiza un seguimiento del movimiento del cuerpo humano en los ejes x, y, z.

- **¿Dispositivo que estamos usando?**

  Acelerómetro Wocket (+ - 4g, frecuencia de muestreo = 90Hz)



# Introduccion

- El acelerómetro Wocket contiene un acelerómetro triaxial, un microprocesador, un transmisor Bluetooth y una batería recargable.
- Estos son lo suficientemente pequeños y se pueden usar cómodamente en todas las ubicaciones del cuerpo al mismo tiempo.
- Los datos sin procesar del acelerómetro se adquieren y se envían mediante bluetooth a un teléfono inteligente.


Wocket listo para ser colocado en el cuerpo

# 2

# Motivacion

# Motivacion

- Analizar y comprender el proceso de los dispositivos portátiles comerciales.
  - Los sistemas de reconocimiento de actividad física disponibles comercialmente como Fitbit, Nike + FuelBand, etc. se utilizan ampliamente, pero su algoritmo no ha sido validado, es decir, sigue siendo un sistema de caja negra.
  - En este proyecto, informamos nuestros esfuerzos para reconocer las actividades humanas trabajando en datos similares de acelerómetro sin procesar.
  - Al hacerlo, obtenemos una comprensión profunda del sistema de clasificación de actividades y proporcionar recomendaciones basadas en nuestros hallazgos.

# Motivacion

- Aplicaciones de estos dispositivos en industrias como:
  - Salud: seguimiento de la actividad física, control de la salud, detección de caídas.
  - Social: comparta sus actividades de acondicionamiento físico en sitios de redes sociales como Facebook, etc.
  - Estilo de vida: comportamientos sensibles al contexto.
  - Publicidad dirigida: publicidad basada en las actividades del usuario.
  - Gestión y Contabilidad Corporativa.

# 3

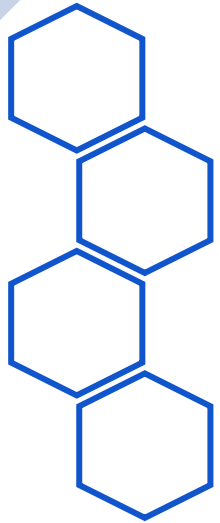
# Objetivos

# Objetivos

- Clasificar de las actividades diarias del usuario mediante el análisis y el procesamiento de datos del acelerómetro de wocket.
- Sugerir la mejor posición posible para la ubicación del sensor según la precisión.
- Sugerir la mejor combinación de sitios de colocación de sensores para clasificar las actividades.


# 4

# Definición del Problema



School of Computer Science
UNSA

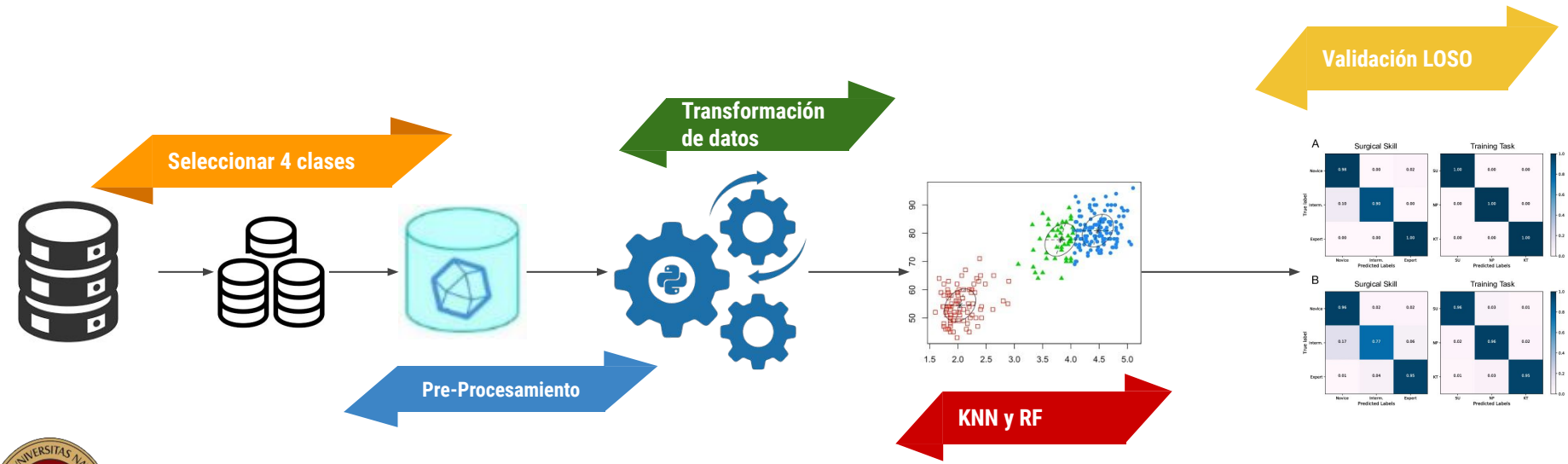
# Definición del Problema

- Estamos trabajando en este proyecto para reconocer algunas de las actividades humanas cotidianas más importantes:
  - Caminando
  - Ciclismo
  - Acostado boca arriba
  - Sentado
- Los acelerómetros se colocan en cinco lugares del cuerpo al mismo tiempo.
  - Brazo dominante
  - Muñeca dominante
  - Cadera dominante
  - Muslo dominante
  - Tobillo dominante
- Estos sitios de colocación fueron seleccionados debido a su relevancia en la investigación de monitoreo del ejercicio.

# 5

# Pasos de Minería de Datos

# Proceso General



# Data Set

# Dataset

Tenemos un conjunto de datos sin procesar de 33 participantes. Para cada participante tenemos los siguientes archivos:

- ❏ 5 archivos son los diferentes archivos de salida de cada sensor colocado en diferentes lugares.
- ❏ El sexto archivo es el archivo de intervalo de anotación (AnnotationIntervals.csv).

Los archivos antes mencionados son:
• AnnotationIntervals.csv
• Wocket_00_DatosDominantesCorregidos.csv
• Wocket_01_DatosRefutados_MusloDominante.csv
• Wocket_02_DatosRayosCorregidos_CaderaDominante.csv
• Wocket_03_DatosRayosCorregidos_MuñecaDominante.csv
• Wocket_04_DatosDominantesCorregidos.csv

Los archivos AnnotationIntervals.csv tienen los intervalos de tiempo y las anotaciones de actividad y los archivos wocket tienen la aceleración x,y,z junto con la marca de tiempo.

# Dataset

Demographics: 33 participants

11 Varones

22 Mujeres

Edad :18-75

Altura: 168.5 +/- 9.3cm

Peso: 70.0 +/- 15.6 kg

| STARTTIME | ENDTIME | activity |
|---|---|---|
| 12-02-2009 10:38 | 12-02-2009 10:39 | sitting |
| 12-02-2009 10:39 | 12-02-2009 10:40 | cycling:-70-rpm_-50-watts_-.7-kg |
| 12-02-2009 10:40 | 12-02-2009 10:43 | walking:-natural |
| 12-02-2009 10:43 | 12-02-2009 10:45 | lying |

Sample Annotations.csv

| Time Stamp | X | Y | Z |
|---|---|---|---|
| 1.25975E+12 | 427 | 434 | 434 |
| 1.25975E+12 | 486 | 510 | 420 |
| 1.25975E+12 | 481 | 477 | 423 |
| 1.25975E+12 | 475 | 490 | 422 |

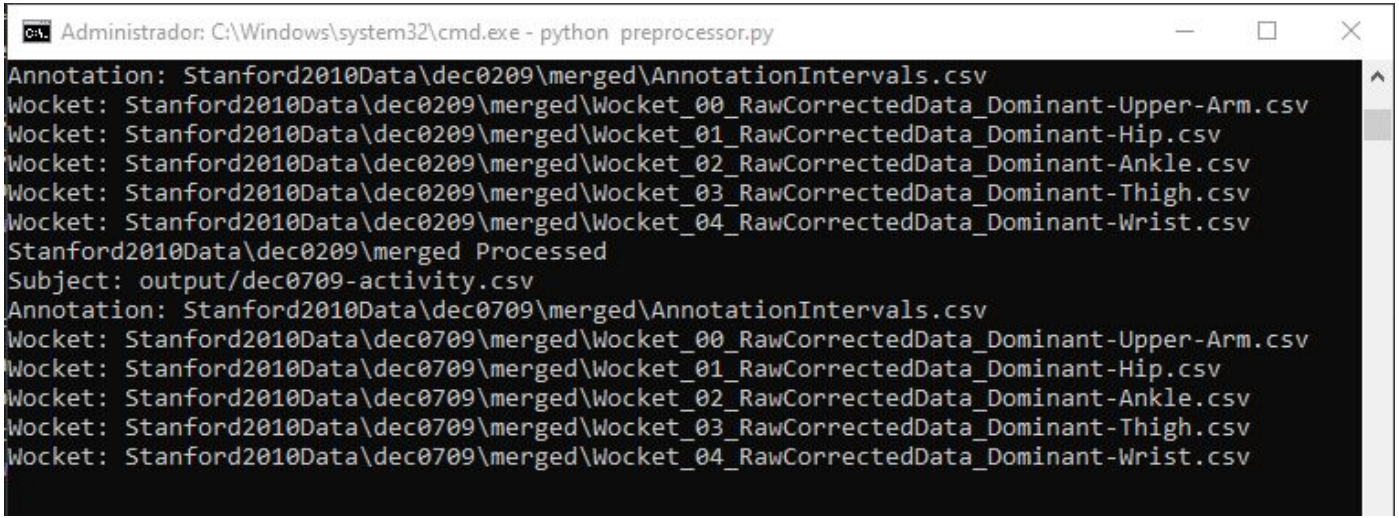
Sample Wacket.csv

# PreProcesamiento

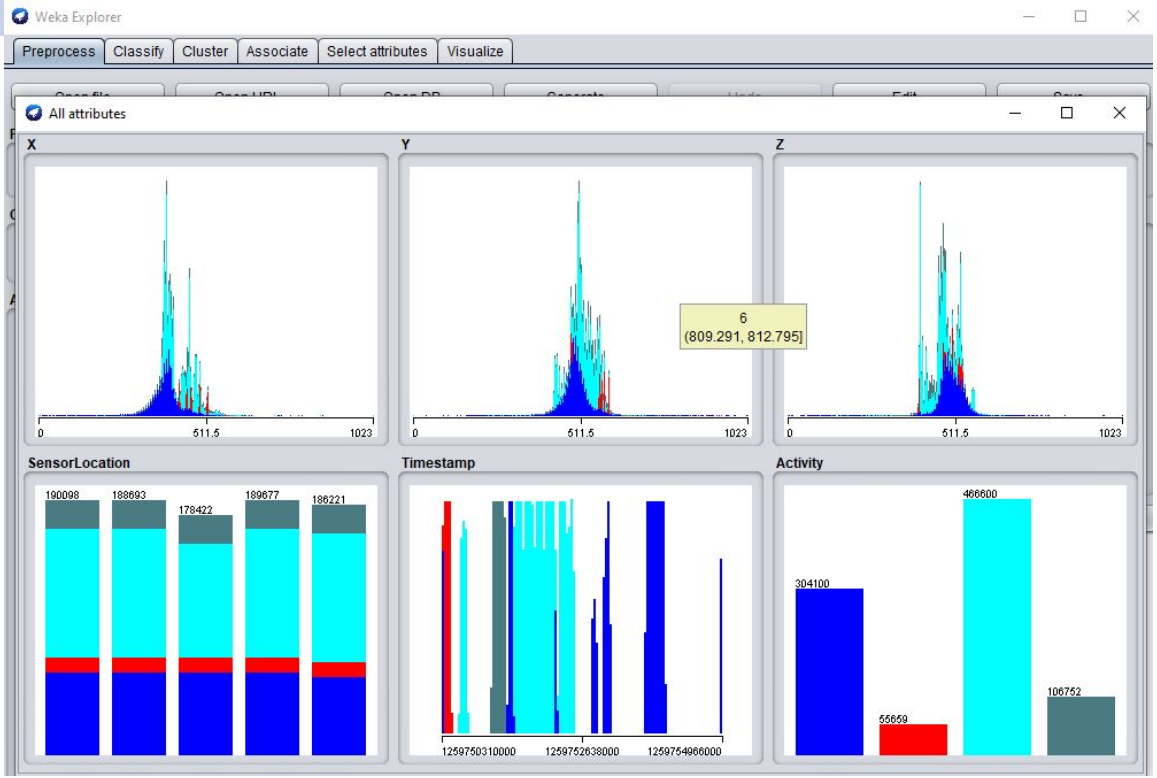
# Preprocesamiento

Para el preprocesamiento, se extrajo los archivos mencionados de cada participante y se creó un archivo combinado. Este archivo combinado se crea para cada participante. Para la fusión, tomamos cada archivo wocket con la marca de tiempo y lo fusionamos con el archivo de anotación colocando cada marca de tiempo en el intervalo de tiempo correcto. En el archivo de anotaciones añadimos 2 segundos a la hora de inicio y restamos 2 segundos a la hora de finalización para evitar la fase de transición entre dos actividades. La fase de transición se descartó para que los datos resultantes fueran lo más inequívocos posible en cuanto a su etiqueta de actividad. El archivo final fusionado tiene las columnas X,Y,Z, Ubicación del sensor, Sello de tiempo y Actividad. Esto se realizó para los datos de los 33 participantes.

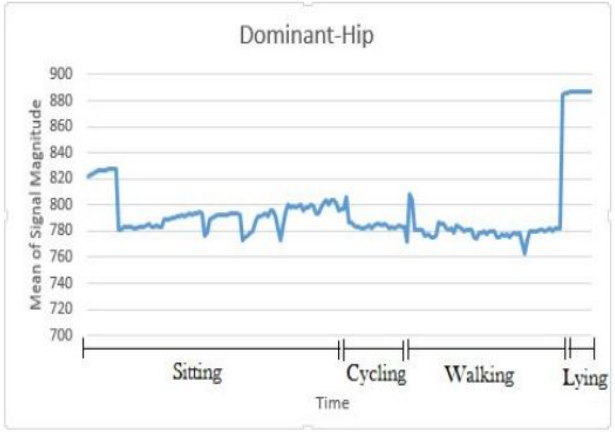
# Preprocesamiento

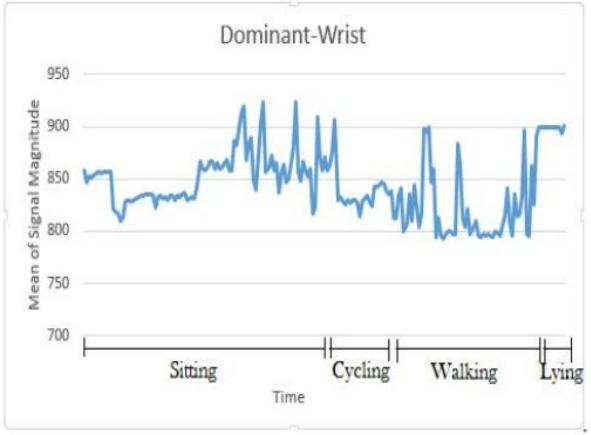


# Data

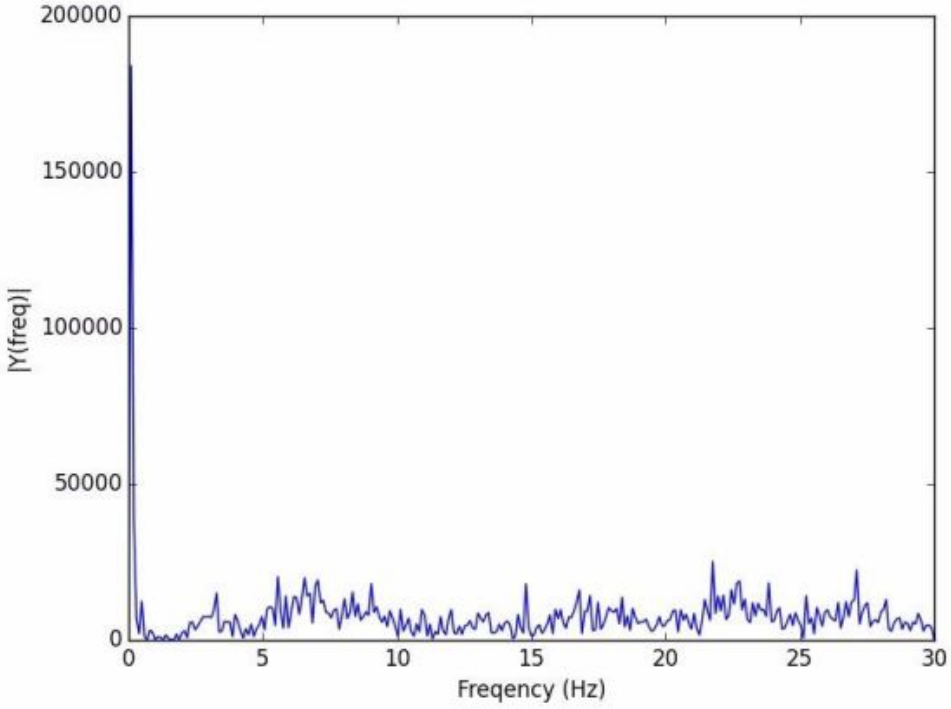

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
All attributes
X
Y
Z
6
(809.291, 812.795]
0
511.5
1023
SensorLocation
190098
188693
178422
189677
186221
Timestamp
1259750310000
1259752638000
1259754966000
Activity
304100
55659
466600
106752

# Magnitud de señal media para cada sitio de sensor

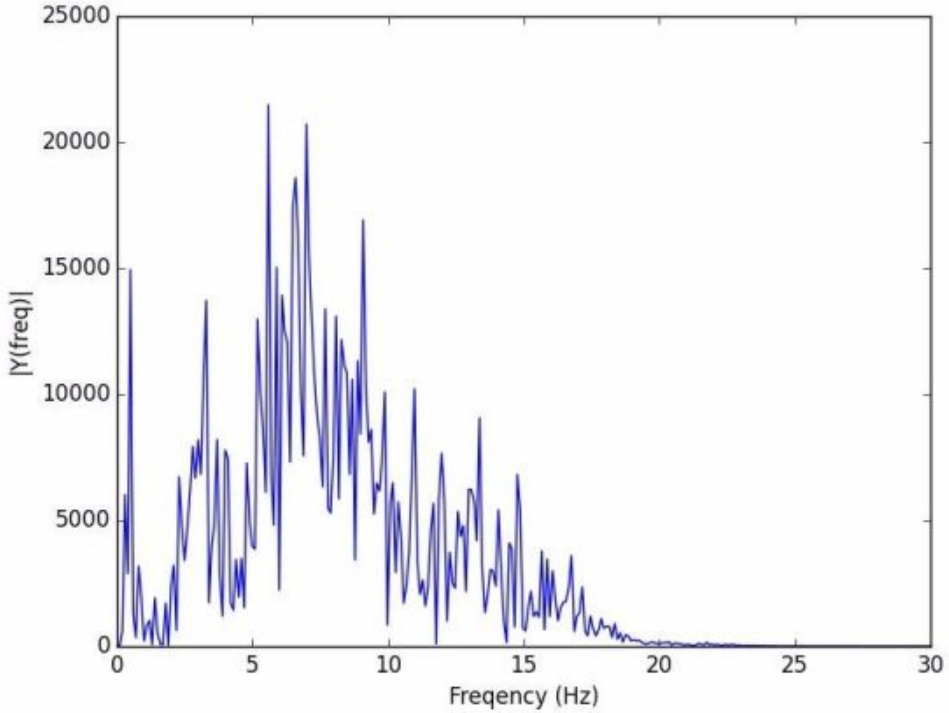



# Magnitud de señal media para cada sitio de sensor




## Frecuencia antes del Filtrado

# Magnitud de señal media para cada sitio de sensor




## Frecuencia después del Filtrado

# Algoritmos

# Patrones KNN y Random Forest (Usando Gridsearch)

KNN (K-Nest-Neighbors) y RF (Random Forest) fueron los algoritmos usados en el desarrollo del proyecto, debido a que demostraron ser los más óptimos para la clasificación de los datos.

# Algoritmo KNN

KNN es un algoritmo de aprendizaje supervisado, cuyo aprendizaje está basado en instancias o aprendizaje perezoso, en el cual la función sólo se aproxima localmente y todo el cálculo se aplaza hasta la parte de la clasificación, es uno de los algoritmos más sencillos y con menor complejidad O(n).

# Algoritmo KNN

```
import numpy as np
import pandas as pd
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.grid_search import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
```

# Algoritmo KNN

Se puede obtener un valor óptimo de K mediante varias técnicas heurísticas, por ejemplo, la validación cruzada. Y también con el uso del método Gridsearch.

# Algoritmo KNN

```
knn_wrist = KNeighborsClassifier(n_neighbors=3, algorithm='auto', weights='uniform')
param_grid = {
    'n_neighbors': range(3,11,2),
    'algorithm': ['auto', 'ball_tree','kd_tree'],
    'weights': ['uniform','distance'],
}
knn_wrist_gs = GridSearchCV(knn_wrist, param_grid=param_grid)
knn_wrist_gs.fit(train_data_wrist, target_label_wrist)
predicted = knn_wrist_gs.predict(test_data_wrist)
print "Best parameters to be used for training the model",knn_wrist_gs.best_params_
print "\n"
```

# Algoritmo KNN

```
print "Beginning 10-Fold cross validation on all the 33 subjects for sensor at wrist position..."
clf_cross_val = KNeighborsClassifier(n_neighbors=9, algorithm='auto', weights='uniform')
scores = cross_val_score(clf_cross_val, train_data_wrist,target_label_wrist,cv=10)
print "Scores for each fold:"
print scores
print "--------------------------------------------------------------------------------"
print ("Accuracy for 10Fold cross validation using KNN: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
print "\n"
```

# Algoritmo KNN

KNN fue usado en la clasificación de las actividades anteriormente mencionadas además de la lectura y clasificación de los datos captados por los sensores de la cintura, del brazo y otros sensores ubicados en otras áreas del cuerpo de los participantes, también participó en la lectura de los datos de entrenamiento.

# Algoritmo RF

- ★ Los bosques aleatorios están basado en los árboles de decisión y es un algoritmo de aprendizaje por conjuntos para la clasificación, que funcionan construyendo múltiples árboles de decisión
- ★ RF es veloz y se ejecuta de manera eficiente en grandes bases de datos, además los bosques generados pueden reusarse en otros datos sin tener que ajustarse en exceso y su complejidad es O(M(mn log n)), siendo n las instancias, m los atributos y M el número de árboles creados.

# Algoritmo RF

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.grid_search import GridSearchCV
```

# Algoritmo RF

RF crea múltiples árboles de clasificación, con el fin de clasificar un nuevo objeto a partir de un vector de entrada, este vector es colocado en cada árbol del bosque, cada árbol da una clasificación sobre el vector y se puede decir que el árbol votó por esa clase, de esa forma el bosque elige la clasificación que tiene más votos en todo el bosque.

# Algoritmo RF

```
# clf = RandomForestClassifier(n_estimators=100, criterion='entropy')
# clf.fit(train_data_wrist, target_label_wrist)
# predicted = clf.predict(test_data_wrist)
rfc_wrist = RandomForestClassifier(n_estimators=100, criterion='entropy',n_jobs=-1)
param_grid = {
    'n_estimators': [50, 100, 200],
    'criterion': ['entropy', 'gini']
}
rfc_wrist_gs = GridSearchCV(rfc_wrist, param_grid=param_grid)
rfc_wrist_gs.fit(train_data_wrist, target_label_wrist)
predicted = rfc_wrist_gs.predict(test_data_wrist)
# print "Best parameters to be used for training the model",rfc_wrist_gs.best_params_
```

# GridSearch

Gridsearch ayuda a restringir el rango de valores necesarios para los algoritmos anteriormente descritos, esta restricción de rango ayuda a poder limitar los valores necesarios y por lo tanto optimizar los resultados.

# GridSearch

Gridsearch aceptó un rango de parámetros que fueron usados para ajustar los datos y este a su vez puede ser optimizado por la búsqueda cruzada, usando ambos métodos se llegó a obtener los siguientes rangos de valores para KNN y RF.

# GridSearch

- KNN:

  - El rango de K es [9-11] y el peso es uniforme para todos los puntos.

  - Cálculo de las distancias: euclidiana.

- RF:

  - Rango de árboles de decisión generados es [50-200].

  - Algoritmo empleado: InfoGain (es una entropía).

# 6

# Resultados y Evaluación

# Resultados

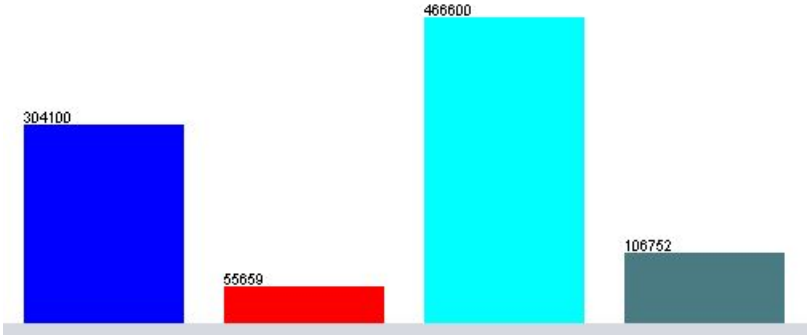
# Visualización

**Selected attribute**

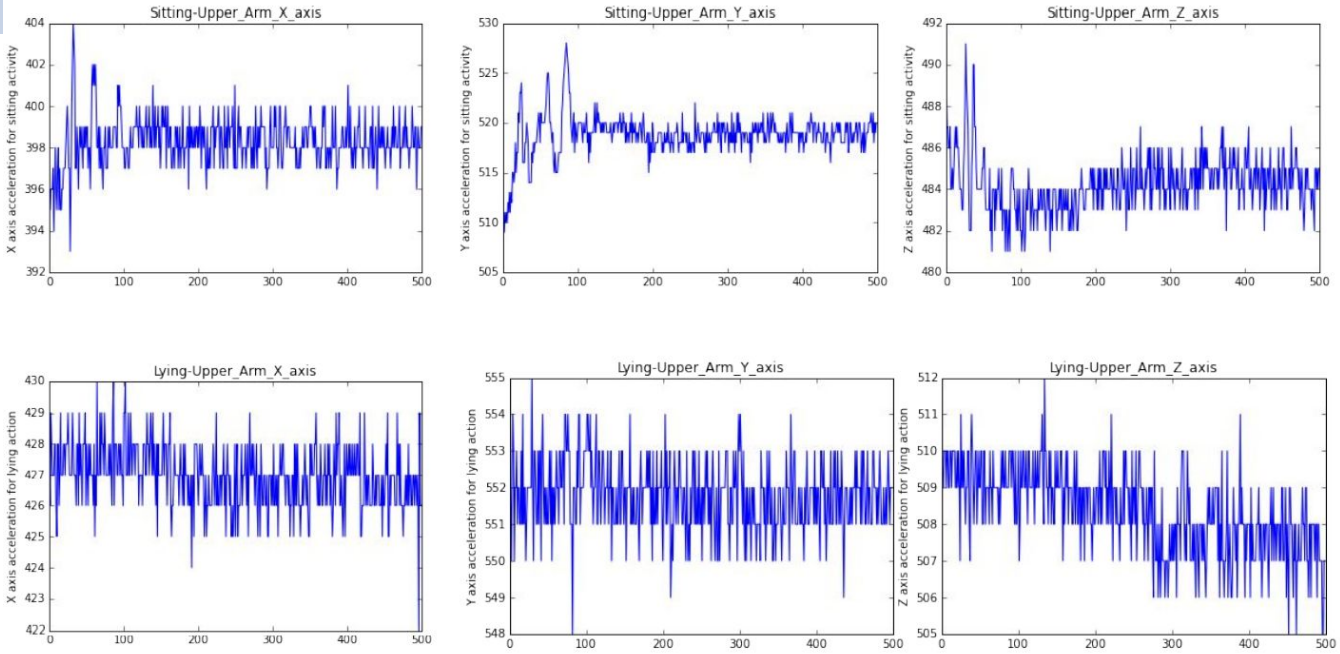
Name: Activity  Type: Nominal
Missing: 0 (0%)  Distinct: 4  Unique: 0 (0%)

| No. | Label | Count | Weight |
|---|---|---|---|
| 1 | walking:-natural | 304100 | 304100.0 |
| 2 | lying:-on-back | 55659 | 55659.0 |
| 3 | sitting:-legs-straight | 466600 | 466600.0 |
| 4 | cycling:-70-rpm_-50-... | 106752 | 106752.0 |

Class: Activity (Nom) ▼  Visualize All



# Resultados




Trazado de los valores de aceleración x, y, z para actividades sentado y acostado para la posición del sensor en la parte superior del brazo

# Evaluacion

# Evaluacion

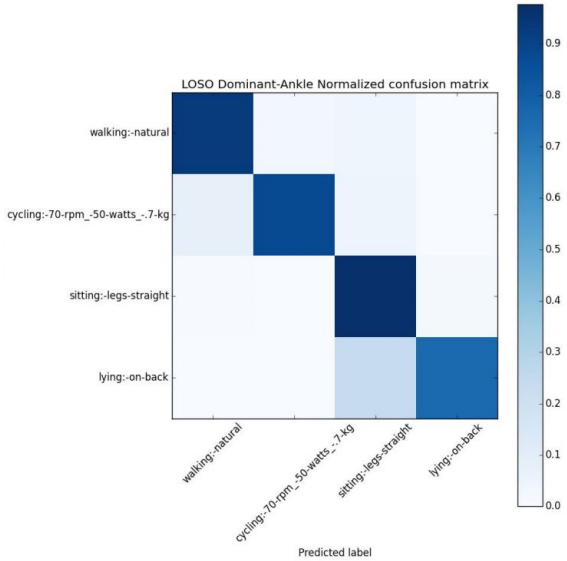
Usamos los siguientes métodos para evaluar nuestro modelo:

- Validación cruzada de 10 veces
- Leave-One-Subject-Out (LOSO): simula una situación de la vida real

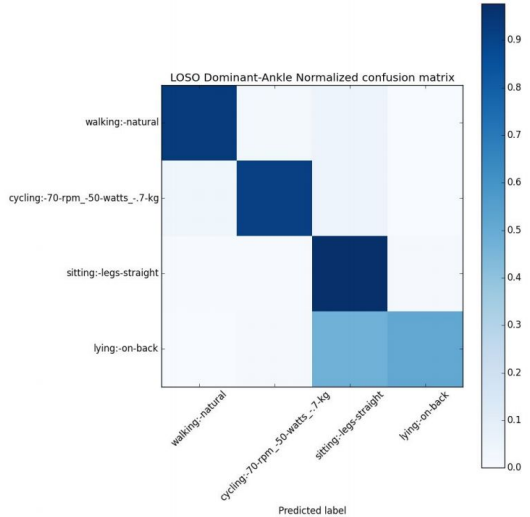
Las medidas de resultado incluyeron Exactitud, Precisión, Recuperación y puntuación F1.

# Evaluacion

Matriz de confusión después de LOSO CV basada en datos de muñeca :



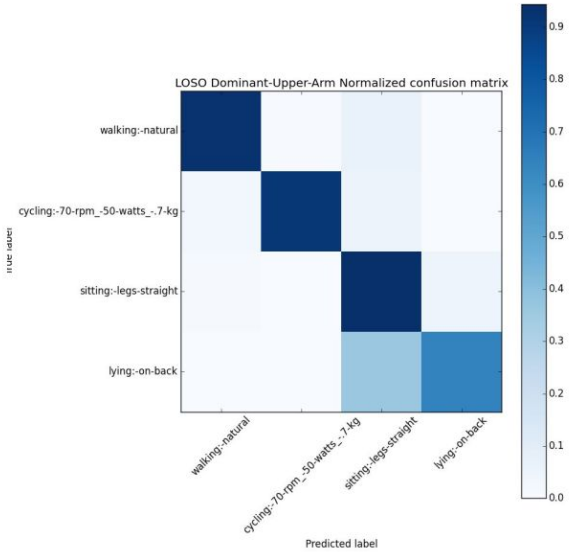

Random Forest



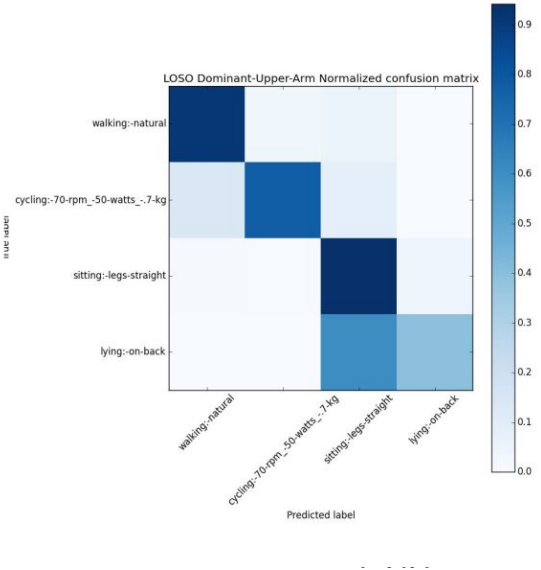

k-NN

# Evaluacion

Matriz de confusión después de LOSO CV basada en los datos del brazo dominante superior:



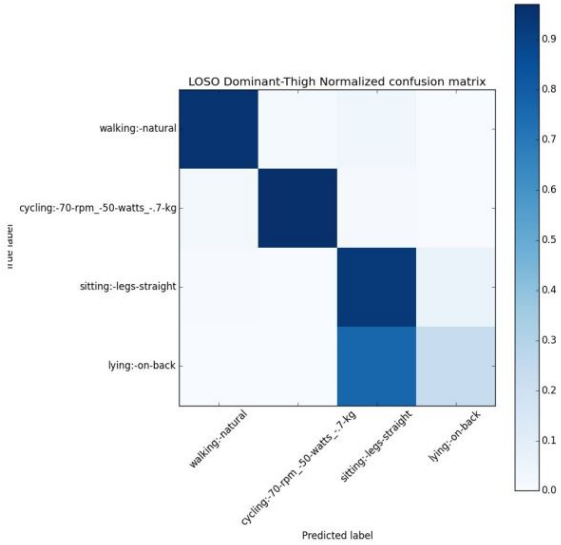

Random Forest



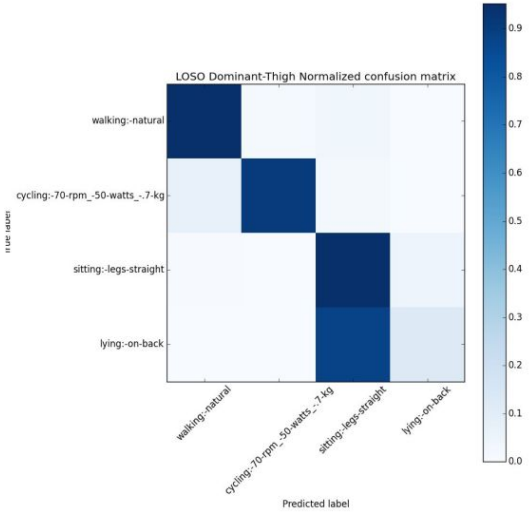

k-NN

# Evaluacion

Matriz de confusión después de LOSO CV basada en datos del muslo :



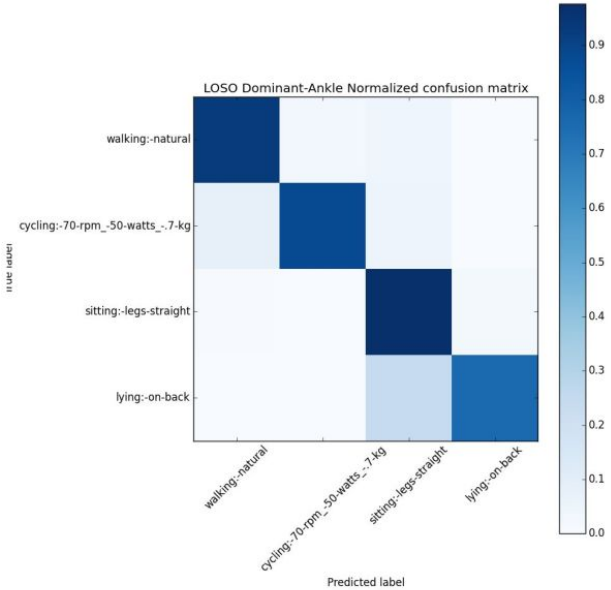

Random Forest



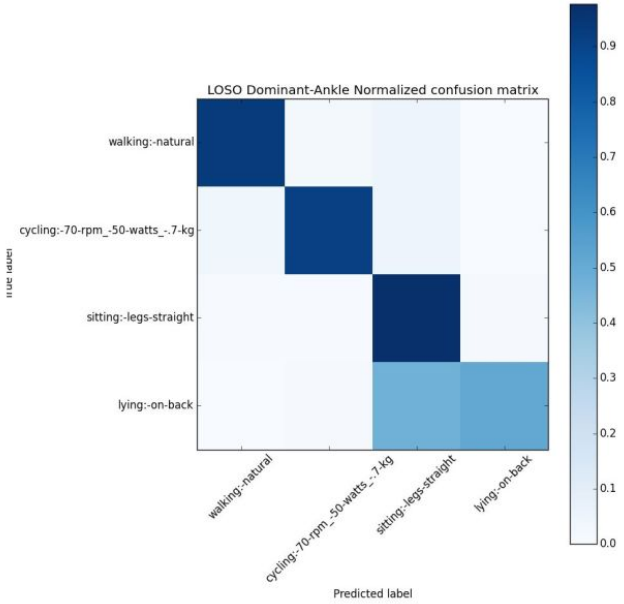

k-NN

# Evaluacion

Matriz de confusión después de LOSO CV basada en datos de tobillo :



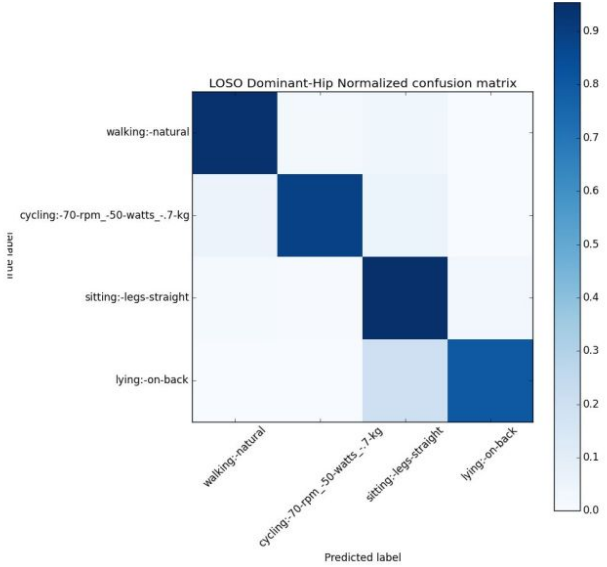

Random Forest



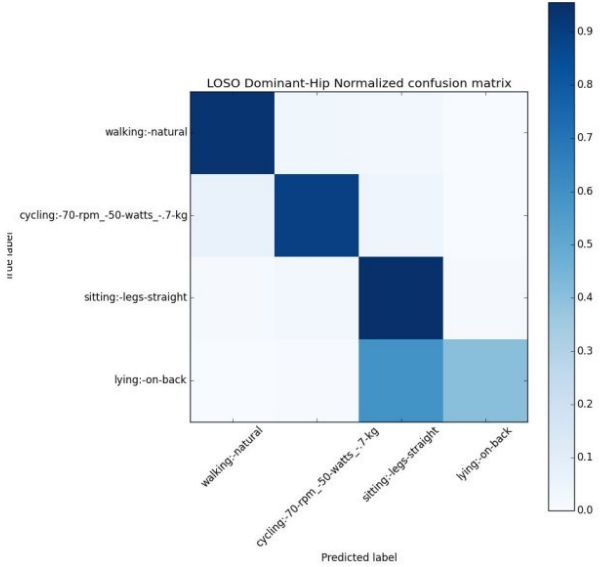

k-NN

# Evaluacion

Matriz de confusión después de LOSO CV basada en datos de Cadera :




Random Forest




k-NN

# Evaluacion

| Clasificador | CV | Posicion del Sensor | Caminar ( Walking) | Cycling(Ciclismo) | Sentado ( Sitting) | Lying |
|---|---|---|---|---|---|---|
| Random Forest | LOSO | Muneca | 88.76 % | 80.24 % | 91.89 % | 50.83 % |
| | | Cadera | 93.95 % | 88.47 % | 95.47 % | 80.45 % |
| | | Muslo | 95.52 % | 96.36 % | 93.01 % | 23.93 % |
| | | Tobillo | 92.95 % | 88.86 % | 97.46 % | 75.90 % |
| | | Brazo | 92.95 % | 91.37 % | 94.29 % | 65.32 % |

# Evaluacion

| Clasificador | CV | Posicion del Sensor | Caminar ( Walking) | Cycling(Ciclismo) | Sentado ( Sitting) | Lying |
|---|---|---|---|---|---|---|
| KNN | LOSO | Muneca | 85.58 % | 56.68 % | 85.36 % | 41.12 % |
| | | Cadera | 93.69 % | 89.48 % | 95.50 % | 40.29 % |
| | | Muslo | 95.20 % | 90.84 % | 94.93 % | 11.96 % |
| | | Tobillo | 93.26 % | 91.32 % | 97.67 % | 51.40 % |
| | | Brazo | 91.37 % | 77.59 % | 94.22 % | 39.05 % |

# Evaluacion

| Clasificador | CV | Posicion del Sensor | Caminar ( Walking) | Cycling(Ciclismo) | Sentado ( Sitting) | Lying |
|---|---|---|---|---|---|---|
| Random Forest | 10 Fold | Muneca | 87.93 % | 78.04 % | 90.98 % | 47.73 % |
| | | Cadera | 94.97 % | 91.39 % | 95.45 % | 80.46 % |
| | | Muslo | 95.55 % | 98.19 % | 91.18 % | 25.69 % |
| | | Tobillo | 95.07 % | 86.26 % | 98.07 % | 75.44 % |
| | | Brazo | 94.14 % | 94.53 % | 94.78 % | 60.73 % |

# Evaluacion

| Clasificador | CV | Posicion del Sensor | Caminar ( Walking) | Cycling(Ciclismo) | Sentado ( Sitting) | Lying |
|---|---|---|---|---|---|---|
| KNN | 10 Fold | Muneca | 83.27 % | 60.44% | 83.58 % | 36.93 % |
| | | Cadera | 94.17 % | 89.87 % | 95.74 % | 45.86 % |
| | | Muslo | 95.54 % | 92.40 % | 92.54 % | 16.02 % |
| | | Tobillo | 93.85 % | 90.63 % | 96.58 % | 53.65 % |
| | | Brazo | 90.78 % | 79.52 % | 92.93 % | 39.78 % |

# Evaluacion

- Combinando datos de los dos sitios de colocación principales, para este caso Cadera y Tobillo.

| | Caminar ( Walking) | Cycling(Ciclismo) | Sentado ( Sitting) | Lying |
|---|---|---|---|---|
| **Precision** | 0.95429058 | 0.899217 | 0.956576 | 0.879931 |
| **Recordar** | 0.9537383 | 0.887182 | 0.979197 | 0.874007 |
| **F1** | 0.95011728 | 0.887496 | 0.966129 | 0.862288 |
| **Accuracy** | 0.953738298 | 0.887182 | 0.979197 | 0.90132 |

- Por lo tanto, podemos ver en la tabla anterior que la combinación de datos de Cadera y Tobillo mejoró las medidas de resultado generales.

# 7

# Conclusiones

# Conclusiones

- Características predictivas: Media, Desviación Estándar,  Mediana, junto con características del dominio de frecuencia extraídas del vector de magnitud de la señal.
- Tanto para LOSO como para Cross Validation de 10 iteraciones,  RF se comportó mejor que k-NN.
- Los datos de la cadera dominante fueron los más discriminativos en la clasificación, seguidos por el tobillo dominante.
- Recomendamos la combinación de ambos sitios para mejorar la clasificación.

# GRACIAS