a) **Crear archivo nuevo Python:** Se puede utilizar cualquier editor de texto o IDLE para Python, por ejemplo, Spyder.

b) **Importar librerías:** Para lograr los objetivos es necesario importar las siguientes librerías de python**:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.utils import shuffle
```

c) **Cargar datos:** Para leer los datos de un archivo .csv en Python es necesario:

```
separador="," ## declarar el separador que tiene el archivo
f = open("7posiciones.csv", "r")  # abrir el archivo
datos = []
for x in f:
        datos.append(x.split(separador)) # anexar cada línea del  archivo a una lista
```

d) **Longitud:** Es necesario saber cuantos datos tiene el archivo, para eso pasar la lista a un arreglo de numpy y utilizar la función: shape[0].

e) **Constantes:** De acuerdo con la organización del archivo donde guardan los datos del acelerómetro es necesario declarar las siguientes constantes:

```
COLUMNA_FECHA = 0
COLUMNA_PAQUETE = 1
COLUMNA_CLIENTE = 2
COLUMNA_X1 = 3
COLUMNA_X2 = 4
COLUMNA_X3 = 5
COLUMNA_X4 = 6
COLUMNA_X5 = 7
COLUMNA_X6 = 8
COLUMNA_LABEL = 9
feat_labels = [ "X1" , "X2", "X3" , "X4", "X5", "X6"]
```

f) **Features vs Labels:** Es necesario dividir los datos en una matriz de caracteristicas, y un vector de etiquetas. Así cada medición queda asociada a un estado especifico.

```
Xstring = datosnp[:,COLUMNA_X1:COLUMNA_X6+1]
Ystring = datosnp[:,COLUMNA_LABEL]
```

g) **From String to double:** Al leer los datos, se interpretan como String, es necesario cambiar el tipo recorriendo todo el arreglo y cambiando el tipo de dato. A continuación, se muestra el recorrido doble sobre una matriz, queda por realizar el recorrido sencillo sobre el vector:

```
for datos in Xstring:
        for dato in datos:
                fila.append(float(dato))
        X.append(fila)
        fila=[]
```

h) **Numpy array:** Para utilizar la librería es necesario que tanto la matriz de características como el vector de etiquetas sean arreglos de numpy. Para ello utilizar la función *np.array()*

i) **Dividir datos de entrenamiento y de prueba:** Al entrenar modelos de ML es necesario siempre realizar una partición de la información. Utilizar un porcentaje para entrenar y otro porcentaje para validar. Para ello, se utiliza la función *train_test_split*. Además, es recomendable realizar una gráfica de la distribución de los datos de entrenamiento, para así saber si es uniforme la muestra:

```
#### Reorganizar aleatoreamente
X, Y = shuffle(X,Y)
#### Disvisión entre datos de entrenamiento y de prueba. 70, 30.
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=0)
```

j) **Realizar entrenamiento:** Para ello se utilizará la función *.fit()* después de crear el modelo con la función *RandomForestClassifier(),* tal y como se muestra a continuación:

```
model_rf = RandomForestClassifier(n_estimators=100, max_features=4,
min_samples_leaf=10,random_state=0, n_jobs=2)
model_rf.fit(X_train, Y_train.ravel())
```

k) **Validez:** Para determinar que tan preciso es el modelo entrenado se realiza la predicción de los datos de prueba y se compara con los reales:

```
y_pred = model_rf.predict(X_test)
precision=accuracy_score(Y_test, y_pred)
```

l) Una vez realizado todos los pasos anteriores descargar el archivo randomforestPMDI.py con la solución, comparar con la suya, ejecutar y analizar los resultados