**Pratical Machine Learning**

**Import Libraries**

**import** **matplotlib.pyplot** **as** **plt**

**import** **pandas** **as** **pd**

**import** **seaborn** **as** **sns**

**import** **numpy** **as** **np**

**import** **math**

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn** **import** metrics

**from** **sklearn.neighbors** **import** KNeighborsClassifier

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn.metrics** **import** classification\_report

**from** **sklearn.metrics** **import** accuracy\_score

**from** **sklearn.metrics** **import** confusion\_matrix

**import** **warnings**

warnings.filterwarnings('ignore')

# Download Data

### The original training and test data has 160 variables.

### The columns with NA entries have been removed. Five (5) variables were removed.

*# Importing the Dataset*

df = pd.read\_csv('pml-training.csv')

*# Clear all null data*

df.dropna(inplace=**True**)

*# Total rows and columns*

print("Train data line and colum: **{}**".format(df.shape))

Train data line and colum: (406, 160)

# Train Test Split

### We will divide our dataset into training and test splits, which gives us a better idea as to how our algorithm performed during the testing phase.

### This way our algorithm is tested on un-seen data, as it would be in a production application.

*# Preprocessing*

*# The next step is to split our dataset into its attributes and labels*

cols = ['raw\_timestamp\_part\_1',

'raw\_timestamp\_part\_2',

'num\_window',

'roll\_belt',

'pitch\_belt',

'yaw\_belt',

'gyros\_forearm\_x',

'gyros\_forearm\_y',

'gyros\_forearm\_z',

'accel\_forearm\_x',

'accel\_forearm\_y',

'accel\_forearm\_z',

'magnet\_forearm\_x',

'magnet\_forearm\_y',

'magnet\_forearm\_z']

X = df[cols]

y = df.classe

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30,random\_state=0)

print (X\_train.shape)

(284, 15)

print (X\_test.shape)

(122, 15)

# Model

### The first step is to import the DecisionTreeClassifier class from the sklearn.neighbors library.

### In the second line, this class is initialized with one parameter.

### This is basically the value for the K.

### There is no ideal value for K and it is selected after testing and evaluation, however to start out, 5 seems to be the most commonly used value for DECISION TREE algorithm.

### After all the work of data preparation, creating and training the model DECISION TREE regression model and fit the model on the training data.

# Predictions

### It is extremely straight forward to train the DECISION TREE algorithm and make predictions.

*# Tree classifer*

clf = DecisionTreeClassifier()

*# Decision Tree Classifer*

clf = clf.fit(X\_train,y\_train)

*# prever conjunto de testes*

y\_predc = clf.predict(X\_test)

y\_predc

B A B A A E D B A A B C B A E E A B B B

# Evaluating the Algorithm

For evaluating an algorithm, confusion matrix, precision, recall and score are the most commonly used metrics. ACCURACY 0.6311475409836066

**from** **sklearn.metrics** **import** classification\_report, confusion\_matrix

print("CONFUSION MATRIX")

print(confusion\_matrix(y\_test, y\_predc))

print("")

print("METRICS")

print(classification\_report(y\_test, y\_predc))

print("")

*#Cria a árvore de decisão e calcula a curácia*

print("ACCURACY")

**from** **sklearn** **import** tree

clf = tree.DecisionTreeClassifier()

clf = clf.fit(X\_train, y\_train)

clf.score(X\_test,y\_test)

CONFUSION MATRIX

[[22 3 7 6 0]

[ 2 13 6 1 0]

[ 0 2 12 2 0]

[ 2 0 1 14 0]

[ 1 2 1 1 24]]

METRICS

precision recall f1-score support

A 0.81 0.58 0.68 38

B 0.65 0.59 0.62 22

C 0.44 0.75 0.56 16

D 0.58 0.82 0.68 17

E 1.00 0.83 0.91 29

micro avg 0.70 0.70 0.70 122

macro avg 0.70 0.71 0.69 122

weighted avg 0.75 0.70 0.71 122

ACCURACY

0.6311475409836066

**Conclusion**

DECISION TREE is a simple yet powerful classification algorithm.

It requires no training for making predictions, which is typically one of the most difficult parts of a machine learning algorithm.

The DECISION TREE algorithm have been widely used to find document similarity and pattern recognition.

It has also been employed for developing recommender systems and for dimensionality reduction and pre-processing steps.