Pratical Machine Learning

Import Libraries

```
# Importing 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 cross_val_score
from sklearn.model_selection import train_test_split
from sklearn import metrics

from sklearn import tree
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)
```

```
# Columns present in the dataset
print(df.columns)
Index(['Unnamed: 0', 'user_name', 'raw_timestamp_part_1',
       'raw_timestamp_part_2', 'cvtd_timestamp', 'new_window', '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',
'classe'],
      dtype='object', length=160)
```

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. 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. I did the cross validation and predicted a test set.

```
# Cross Validation
print("-- 10-fold cross-validation "
      "[using setup from previous post]")
dt_old = DecisionTreeClassifier(min_samples_split=20,
                                random_state=99)
dt_old = dt_old.fit(X, y)
scores = cross_val_score(dt_old, X, y, cv=10)
print("mean: {:.3f} (std: {:.3f})".format(scores.mean(),
                                          scores.std()),
                                          end="\n\n" )

# predict set of tests
y_predc = dt_old.predict(X_test)
```

```
y_predc
-- 10-fold cross-validation [using setup from previous post]
mean: 0.382 (std: 0.101)

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: 0.8688

```
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("")

# calculates accuracy
print("ACCURACY")
from sklearn import tree

cfl = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
clf.score(X_test,y_test)

accuracy_score(y_test, dt_old.predict(X_test))
```

```
CONFUSION MATRIX
[[29  5  0  3  1]
 [ 0 21  0  0  1]
 [ 0  1 15  0  0]
 [ 0  0  0 17  0]
 [ 1  0  1  3 24]]

METRICS
              precision    recall  f1-score   support

           A       0.97      0.76      0.85        38
           B       0.78      0.95      0.86        22
           C       0.94      0.94      0.94        16
           D       0.74      1.00      0.85        17
           E       0.92      0.83      0.87        29
```

```
   micro avg       0.87      0.87      0.87       122
   macro avg       0.87      0.90      0.87       122
weighted avg       0.89      0.87      0.87       122


ACCURACY: 0.8688524590163934
```

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.