```python
# weka_implementation.py
from weka.classifiers import Classifier
from weka.core.converters import Loader
import weka.core.jvm as jvm
from weka.core.dataset import create_instances_from_lists
from weka.filters import Filter
from weka.core.classes import Random
from weka.classifiers import Classifier, Evaluation, PredictionOutput
import weka.plot.graph as graph
import graphviz
import time
import pandas as pd
import numpy as np
from graphviz import Digraph

# Aideen McLoughlin - 17346123
# Taking in the data file location, and the train/test split proportion
# Build a weka C4.5 implementation using the Python Weka Wrapper API
def build_weka_tree(split):
    jvm.start()

    accuracy, time_to_build = build_weka(split)

    jvm.stop()
    return accuracy, time_to_build

def build_weka(split):
    # Load the data file
    loader = Loader(classname="weka.core.converters.CSVLoader")
    train = loader.load_file('train_data_generated.csv')
    test = loader.load_file('test_data_generated.csv')

    # Store the target values for the test data
    # so that the accuracy of the formula can be checked
    test_target = pd.read_csv('test_data_generated.csv')['style'].values
    # Get the dataset used to train the model,
    # so that we can identify what the key values for the class are.
    # As data is split randomly, we cannot assume it is in [ale, lager,stout] order
    train_classes = pd.read_csv('train_data_generated.csv')['style'].values

    # Set the class to be column 3 - the style column
    train.class_index = 3

    # Set the class to be column 3 - the style column
    test.class_index = 3

    # Store the time before starting to build the tree
    starttime = time.time()

    # initialise the time_to_run and accuracy to 0
    time_to_run = 0
    accuracy = 0

    # Build and Train the weka tree
    cls = Classifier(classname="weka.classifiers.trees.J48")

    # Check that the data is valid
    # If so, Build and Train the weka tree
    if len(list(np.unique(train_classes))) != 1:
        cls.build_classifier(train)
        graph = cls.graph
        # Store the time once the tree has been built
        endtime = time.time()

        # Create a list to store the predicted values in
        pred = []
        accurate = []
```

```
            # Get the class labels in the order that they were allocated when training the model
            classes = pd.Series(train_classes).drop_duplicates().tolist()

            correct = 0
            total = 0
            # loop through test dataset, incrementing total every time
            # and incrementing count if the predicted value was correct
            for index, inst in enumerate(test):
                total = total +1
                predicted = classes[int(cls.classify_instance(inst))]
                pred.append(predicted)
                act = test_target[index]
                if predicted == test_target[index]:
                    correct = correct+1
                    accurate.append(1)
                else:
                    accurate.append(0)

            # Get the accuracy of the weka implementation
            accuracy = (correct/total)

            # store the results in a csv file - the predicted class and the actual class
            df = pd.DataFrame()
            df['Actual'] =  test_target
            df['Predicted'] = pred
            df['Accuracy'] = accurate
            filename = "results/weka-results-"+str(round(split,2))+"-"+str(round(accuracy,2))+".csv"
            df.to_csv(filename,index=False,header=True)

            time_to_run = endtime-starttime
        # If the data is invalid, create a node to indicate failure
        elif len(train) == 0:
            graph = Digraph('python_tree_implementation')
            graph.node(name='A', label="Fail", shape='box', style='filled')
        else:
            graph = Digraph('python_tree_implementation')
            graph.node(name='A', label=train_classes[0], shape='box', style='filled')

        # Render a png image of the weka tree to display in the PySimpleGUI popup
        g = graphviz.Source(graph)
        g.format = "png"
        g.render('weka-test.gv', view=False)
        return round(accuracy*100, 2), time_to_run
```