```
# Appendix C9 - measure_calculator.py

class MeasureCalculator():
    def __init__(self):
        self.measures = {
            "BernoulliNB": {
                "cross score": [],
                "cross variance": [],
                "precision": [],
                 "recall": [],
                 "f1": []
            },
            "MultinomialNB": {
                "cross score": [],
                "cross variance": [],
                "precision": [],
                "recall": [],
                "f1": []
            }
        }

    def add_measures(self, cross_validation_measures, accuracy_measures):
        measures = self.combine_measures(cross_validation_measures, accuracy_measures)
        for algo_type, results in measures.items():
            for result, value in results.items():
                self.measures[algo_type][result].append(value)

    def combine_measures(self, cross_validation_measures, accuracy_measures):
        current_measures = {}
        current_measures["BernoulliNB"] = dict(
            list(cross_validation_measures["BernoulliNB"].items()) +
            list(accuracy_measures["BernoulliNB"].items())
        )
        current_measures["MultinomialNB"] = dict(
            list(cross_validation_measures["MultinomialNB"].items()) +
            list(accuracy_measures["MultinomialNB"].items())
        )
        return current_measures

    def averaged_measures(self):
        for algo_type, results in self.measures.items():
            print(algo_type + ":")
            cross_score = (sum(results["cross score"]) / len(results["cross score"]))
            cross_precision = (sum(results["cross variance"]) / len(results["cross varian
ce"]))

            # Print out average of cross eval measure along with its variance
            print("Cross evaluation accuracy: %1.3f (+/- %1.3f)" % (cross_score, cross_pr
ecision))
            results.pop("cross score")
            results.pop(("cross variance"))

            for result, values in results.items():
                # Print out averages of all remaining measures
                print("%s: %1.3f" % (result, (sum(values) / len(values))))

```