Решение домашней работы содержит следующие модули:

- *algorithm0.py* – реализация простейшего алгоритма классификации из задания: объект классифицируется положительно, если каждое его пересечение с объектами из положительного контекста не вкладывается в описания из отрицательного контекста (и наоборот).
- *algorithm1.py* – реализация алгоритма классификации, основанного на голосовании: каждый из положительных объектов “голосует” за положительный результат, если его пересечение с тестируемым объектом не вкладывается в описания из отрицательного контекста (и наоборот).
- *algorithm2.py* – реализация алгоритма классификации, считающего для каждого примера его поддержку в положительном и отрицательном контексте (по формуле из задания). Выбирается класс, соответствующий контексту с большей поддержкой.
- *algorithm3.py* – модификация algorithm2, в которой из поддержки для каждого примера вычитается доля примеров, фальсифицирующих положительную/отрицательную гипотезу.
- *algorithm4.py* – реализация алгоритма классификации, помещающего примеры как положительные/отрицательные при преодолении их поддержки в соответствующих контекстах некоторого порога C (мы пробовали C = 0.5 и C = 0.75).
- *kfold.py* – реализация кросс-валидации с настраиваемым числом параметров K (2 <= K <= 10).
- *utils.py* – реализация вспомогательных методов: загрузка данных, вычисление accuracy/precision/recall/F1-score.

Ниже приведены оценки каждого из пяти описанных выше алгоритмов на тестовых данных. В связи с вычислительной требовательностью этих алгоритмов для получения результатов ниже было использовано подмножество тренировочных/тестовых данных (train1.csv и test.csv):

	Accuracy (с учетом contradictory)	**Precision**	**Recall**	**F1 score**
Алгоритм 0	0.11	1.0	1.0	1.0
Алгоритм 1	1.0	1.0	1.0	1.0
Алгоритм 2	0.54	0.66	0.61	0.63
Алгоритм 3	0.66	0.66	1.0	0.79
Алгоритм 4	0.0	0.0	0.0	0.0

Лучшие результаты на этих данных, таким образом, показал алгоритм 1 (с голосованием).