Домашняя работа содержит следующие алгоритмы:

1. **lazy_fca.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).
2. **fca_votes.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).
3. **supp_fca.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.
4. **supp_falsifiab.py** - реализация алгоритма классификации, в которой из поддержки для каждого примера вычитается доля примеров, фальсифицирующих положительную/отрицательную гипотезу.
5. **antisupp_fca.py** - реализация алгоритма классификации, считающего для каждого примера его поддержку в чужом контексте. Выбирается класс, соответствующий контексту с большей поддержкой.
6. В файле **main.py** происходит основное действие, вызов всех реализованных методов. ***K*** для кросс валидации как второй параметр для каждой функции. К примеру:
supp_falsifiability(df_for_test, 2) здесь K = 2
7. **data_preparation.py** – шкалирование данных, разбиение на классы
8. **estimation.py** – расчет accuracy, recall, precision

	Accuracy
lazy_fca	0.2002
fca_votes	0.7838
supp_fca	0.7621
supp_falsifiab	0.7703
antisupp_fca	0.7979
SVM	0.8202

В итоге SVM оказался самым точным.