**Подготовка белка:**

Для молекулярного докинга был использован белок 3OY3, загруженный с сайта RCSB PDB. Далее, с помощью MGLTools была проведена предобработка структуры: удалены дублирующая цепь B, кристаллизованный лиганд и вода. Растворителей в структуре не оказалось.

Проверка наличия пропусков в аминокислотной последовательности проводилась через SwissModel после перевода pdb в формат FASTA (https://swissmodel.expasy.org), однако в данном случае заполнение пропусков не потребовалось. Если бы они присутствовали, их корректировка могла бы изменить нумерацию аминокислот, поэтому ее пришлось бы восстанавливать в PyMOL.

Добавление атомов водорода выполнено с помощью сервиса H++ (http://biophysics.cs.vt.edu/H++), при этом использовано значение pH = 7.4. Этот уровень pH соответствует физиологическим условиям (клеточная среда), что критично для адекватного распределения протонных состояний аминокислот и, соответственно, корректного моделирования электростатического взаимодействия лигандов с рецептором.

При обработке структуры была выполнена мержинг (объединение) неполярных водородов в MGLTools. Это необходимо для упрощения вычислений и уменьшения количества степеней свободы, что ускоряет молекулярный докинг без значительной потери точности. Расчет зарядов для белка выполнен по методу Колмана. Данный метод применяется в AutoDock и позволяет корректно учитывать электростатику большой молекулы. Рецептор был сохранен в формате pdbqt [**receptor.pdbqt**].

Для настройки области докинга использовался AutoGrid. Параметры сетки (grid box) заданы так, чтобы центр совпадал с положением исходного кристаллического лиганда, поскольку это наиболее вероятный сайт связывания. Докинг по всему белку был признан неэффективным, так как требует значительных вычислительных ресурсов и снижает точность предсказания. Spacing сетки установлен на 0.375 Å, что является стандартным компромиссом между точностью и производительностью расчетов.

**Подготовка лигандов:**

Я решила автоматизировать подготовку лигандов, так как вручную обрабатывать большое количество соединений было бы слишком трудно. Для этого я написала скрипт [**mol\_process.py, config.json**], который выполнял последовательную обработку каждой молекулы из SDF-файла [**test\_set.sdf**].

Сначала скрипт загружал молекулу и проверял наличие пропусков в структуре. Обычно считается, что лиганды – это небольшие небелковые молекулы, но в задании было указано заполнить пропуски в аминокислотной последовательности. Поэтому для небелковых соединений проверка выполнялась с помощью RDKit, а для пептидов использовался Modeller.

После этого происходила оптимизация геометрии молекулы. Обычно я делаю это вручную через Avogadro, но автоматизировать этот процесс с его использованием оказалось затруднительно. Поэтому я использовала RDKit, применяя силовые поля MMFF96 и UFF.

Затем проводилась замена неподдерживаемых атомов. В AutoDock нельзя использовать соединения, содержащие бор или кремний, поэтому такие атомы автоматически заменялись на углерод.

После внесения всех изменений молекула сохранялась в PDB-файл, затем с помощью MGLTools добавлялись атомы водорода (если они терялись на предыдущих этапах), неполярные водороды удалялись.

Заряды рассчитывались методом Gasteiger, так как этот метод более точный и обеспечивает корректный учет электростатики, однако он хуже работает с большими молекулами и ресурсоемкий, поэтому его чаще применяют в случае лигандов.

Обработка торсионных связей выполнялась следующим образом: MGLTools автоматически определял активные торсионы, но если их число превышало 32, то они отключались полностью. Я пыталась реализовать выборочное отключение, но это оказалось сложным из-за разной системы нумерации атомов в RDKit (0-индексация) и MGLTools (1-индексация), что приводило к несоответствиям и ошибкам. Также я пробовала отключать торсионы в уже готовом PDBQT-файле, но AutoDock не принимал такие файлы, вероятно, из-за специфики их структуры. Также иногда ошибки возникали при обработке самим MGLTools, некоторые из них удалось решить, но не все.

Для контроля времени обработки одной молекулы я установила таймаут 10 минут, так как в некоторых случаях оптимизация геометрии занимала слишком много времени. На этом этапе иногда появлялись ошибки, из-за которых вся обработка могла прерываться. В документации не указано, почему это происходит, но, скорее всего, причиной является слишком сложная структура некоторых молекул. Для ускорения процесса скрипт был запущен многопоточно. Итоговые файлы сохранялись в формате PDBQT [**ligands**].

В итоге из 541 молекулы успешно обработаны 497 (92%), остальные не прошли подготовку. Лог всех этапов (успех или неудача) записывался в CSV-файл [**processing-results**]. Вероятно, ошибки можно обойти, но для этого потребуется анализ исходного кода используемых инструментов, так как в официальной документации их причины не описаны.

**Проведение молекулярного докинга**

Для молекулярного докинга использовался AutoDock4 в сочетании с AutoGrid. Из-за большого количества молекул процесс был автоматизирован и распараллелен, чтобы ускорить вычисления. Несмотря на установленный таймаут 1 час, настройки докинга были упрощены для повышения скорости расчетов [**docking.py, config2.json**].

Основной проблемой стало ограничение вычислительных мощностей. Мой ноутбук не справлялся с обработкой такого объема данных, поэтому пришлось задействовать второй ноутбук. В среднем обработка одной молекулы занимала 2-3 минуты, что является очень быстрым временем для докинга, но снижает точность результатов. Для проведения докинга на более высоком уровне требуются соответствующие вычислительные мощности.

Дополнительно возникла проблема с картой электростатического взаимодействия: она содержала только нули. В документации были указаны возможные причины этой ошибки, но ни одна из них не подошла, так как входные файлы были в порядке. Для выяснения причины требуется дополнительное время.

В результате удалось получить 497 докинговых файла (100% от числа подготовленных лигандов).

Однако для некоторых молекул докинг завершился с явно некорректными значениями, например, энергия связывания = 30300000, что является неадекватным результатом. Во-первых, ожидаемые значения должны быть отрицательными, а во-вторых, такая высокая энергия указывает на возможные артефакты – либо молекула не нашла места в активном сайте, либо докинг не завершился корректно, но программа все же выдала результат.

Вероятно, процент успешных докингов можно повысить путем ручной настройки торсионных связей и более детальной калибровки параметров для увеличения точности, однако это потребует значительных вычислительных ресурсов и времени.

**Анализ результатов докинга**

Было автоматизировано внесение данных в таблицу [**binding\_energies.csv**], где фиксировались значения энергии связывания для наилучшего кластера, полученного для каждой молекулы [**table\_after\_docking.py**].

Результаты докинга (DLG-файлы) были преобразованы в PDBQT-формат для дальнейшего анализа с помощью BINANA (<https://durrantlab.pitt.edu/binana/>) [**pdbqt\_files**].

Я пыталась автоматизировать анализ взаимодействий, но BINANA в онлайн-версии не предоставляет возможности пакетной обработки. Для автоматического извлечения данных был написан скрипт, который обрабатывал полученные PDBQT-файлы и извлекал количественные характеристики:

* Число водородных связей
* Количество солевых мостиков
* Число π-π-взаимодействий и т.д.

Однако автоматический анализ позволял получить только числовые показатели, без визуального представления расположения взаимодействий. Для полноценного анализа пространственного расположения связей пришлось использовать веб-версию BINANA вручную, поскольку она предоставляет удобную визуализацию взаимодействий. Скрипт прикрепляю [**binding\_analyzing.py, config3.json**], хотя им так и не воспользовалась, так как связывание 10 молекул я решила оценить и вручную, учитывая описанные выше недостатки скрипта.