Перед нами стояла довольно обычная задача регрессии. Были даны довольно-таки небольшие наборы данных из пары тысяч строк. Я посмотрел значения этих таблиц. По продажам данные у нас до июня, а по клиентам вплоть до сентября. Начинаются в начале 2017 года. Затем я заинтересовался, что за ‘ira’ и не смог загуглить сам провел соответствие между этим параметром и названием локации. Оказалось, что это одно и то же. Так что мы можем выкинуть один столбец без потери точности, а в случае линейной регрессии мы эту точность только приобретем, из-за 100 процентной корреляции между этими двумя колонками. После я заметил, что не представлены данные по клиентам в Village. Затем посмотрел на основные статистики и обратил внимание, что при довольно широком разбросе объема продаж, цена очень плотно сосредоточена у одного значения. Затем я принялся очищать данные от пропущенных значений в ценах и кодировать категориальные переменные, так как без этого модели работать не будут. Перепробовал пару комбинаций самых популярных способов и создал список моделей, который постоянно пополнялся по ходу выполнения кейса. В начале я был уверен, что задача на линейную регрессию, но затем решил ради любопытства добавить дерево решений и результат улучшился в разы. После я вспомнил, что есть ансамбли деревьев, которые должны себя показать еще лучше, и так и оказалось. Причем лучше всего себя показал ExtraTreesRegressor, с которым я не имел дел до этого. Хотя недалеко от него ушел всем знакомый RandomForestRegressor. После я решил ознакомиться с «более продвинутыми библиотеками», среди которых: CatBoostRegressor, LGBMRegressor, XGBRegressor, о них уже давно слышал, а до этого пользовался только sklearn. И как ни странно, они показали результаты хуже, согласно нашей метрики (SMAPE). Также я попробовал и другие более простые алгоритмы по типу ближайших соседей, градиентного спуска и других, и в тоже время многослойный перцептрон, который в принципе можно назвать нейронной сетью. Вычислял перцептрон долго, а прироста также сильного не дал. Возможно, это из-за небольшого количества данных. Так что не все задачи лучше всего просто кормить нейронным сетям. Возвращаясь к метрике, она была не самой стандартной и во встроенной библиотеке ее не было, так что пришлось писать самому. Она призвана быть симметричной, в отличии от того же MAPE, но также дает небольшой скос и штрафует сильнее ошибки в меньшую сторону, то есть лучше предсказывать чуть больше. Что имеет смысл, если мы поставщик и лучше закажем чуть больше, так как конфеты и шоколад портятся не очень быстро, как минимум не за месяц, чем мы закажем меньше и потеряем в прибыли. Мы нашли оптимальный метод кодирования строк и избавления от NaN и при этом также лучшую модель при помощи кросс-валидации и нашей соответственно функции потерь поделив данные на train и test. Где-то, конечно, себя лучше показала линейная регрессия, но она все равно везде сильно уступала ансамблям. Самую простую модель мы получили, теперь можно ее и улучшить. Посмотрели на различные распределения наших переменных, на корреляцию и т. п. Получили, сильную корреляцию между количеством штук в упаковке и ценой, что вполне логично. Для линейной регрессии такого конечно же лучше избегать. Но при этом модель без колонки с ценой работает практически также, что довольно-таки странно и в то же время очень интересно. Далее я попытался добавлять новые колонки или какие-то удалять, но никакого улучшения это не дало. Затем посмотрел на важность наших фич, и там также цена в топе, ну помимо sku, месяца, года и локации. После я решил как-то использовать вторую таблицу с информацией о клиентах. Заполнил данные о клиентах в Village. Смерджил две таблицы. И снова провел обучение. Наша ошибка немного уменьшилась. После чего я решил попробовать оставить только данные о суммарном количестве клиентов, а не кол-ва каждого типа и это тоже дало небольшое улучшение. Ну и, собственно, пришло время предсказаний. Я построил таблицу на три месяца для всех локаций и для всех товаров, о наличии которых у нас есть сведения для соответствующих локаций, так как не для всех локаций и товаров это выполняется. Далее я задумался как заполнять оставшиеся колонки и выяснил, что большая часть однозначно соотносится с sku. Убрал категориальные переменные и заменил пропущенные значения. Получил предсказания, сохранил их в соответствующем виде и перешел к следующему заданию. Его я, вероятно, понял неправильно. А возможно что-то не так с моей моделью, и я использовал вообще не правильный алгоритм. Но у меня предсказания по объему продаж не сильно зависят от наличия переменной с ценой. Возможно это опять же из-за совсем небольшого разброса значения. Может быть стоило как-то стандартизировать наши данные. Уменьшить разброс и избавиться от выбросов. Но при этом увеличение на несколько процентов очень слабо меняет цену.

Будь у меня больше ресурсов и времени, я бы в первую очередь применил больше бы статистики, скалировал, масштабировал, стандартизировал бы различные переменные, избавился бы от выбросов, использовал бы различные критерии, гипотезы. Больше всего сейчас склоняюсь к тому, что проблема все-таки в отсутствии стандартизации. Возможно, и линейная регрессия себя бы гораздо лучше показала. После заняться бы тюнингом нашей модели, так как GridSearchCV или даже RandomizedSearchCV на моей локальной машине работал очень медленно и не дал каких-то сильных улучшений, а colab со своими TPU и GPU также не сильно помог. Также я бы хотел попробовать еще применить нейросеть с keras и tensorflow. Хотя думаю большого улучшения она бы не дала опять же из-за малого количества данных. Еще можно было бы применить stacking, то бишь совместить несколько моделей, дающих лучшие результаты.