**Types of cross selling:**

* Complementary items: what products usually sell together
* Seasonal products
* Data driven: items offered based on historical purchase data for similar shopping behaviors.
* Promoting: pitching the products that are on sale
* Popular items
* Experimentation: the company may try to test what product sells and what doesnt by doing a/b testing
* Impulse: things that people might buy on impulse such as candy bars
* New releases, new version
* Risk products: such as insurance policies being close to expire and check if they want an extension
* Services: offering a subscription based on a one time sale to generate monthly revenue. For example: if someone buys a pdf reader (one time sale), we try to offer a pdf repository on a monthly subscription model

ML notes:

Regression types: Linear regression (univariate) and Multivariate regression.

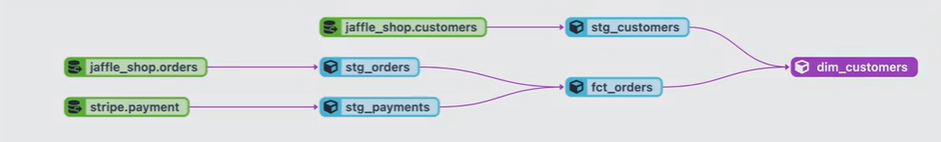
LR:

Squared Error Function = Cost function (J) -> It is so common for LR problems

DBT Notes:

Very important: Using dbt you dont need to write transactional SQL (DDL, DML) to make data models. You just use select statements in SQL.

**Dbt run:** starts building all pieces of a DAG from left to right:



Colors: Green is source, Blue is data models, and Purple is destination

**Dbt test:** will test the models

**DBT docs generate:** will generate the documentation about the model and we can share with the stakeholders

**Dbt build:** builds the model and test it at the same time

Double underscore is very important short key which shows all the keyboard shortcuts.

Each model in dbt has a one-to-one relationship to a table or view in the dwh.

IN order to run a only one model: dbt run - - select <model\_name> (withouth .sql)

If we want to do it recursively for a folder and all its subfolders -> dbt run –select <model\_name>+

To test a source: dbt test - - select source: <source name>

To generate docs: dbt docs generate

Dbt install packages: add a packages.yml file and add the package to it for example:

packages:

- package: dbt-labs/dbt\_utils

version: 1.1.1

Then we say: dbt deps

Seeds: are csv static tables which dont get changed on a regular basis. They are useful for reference data such as country\_codes, etc.

**A/B testing in data science:**

For example we want to know how adding or removing a specific product affects sales. We have categories for products: target market, distance to the warehouse, category, sub-category, expirable or not, brand name, sku, price per unit, popularity score, seasonal, seasion\_name:

Example: Chocolate bar (Berlin, 1.5km, snack, chocolate, Expirable, Lindt, 100gr, 1.2€, 5.5, Yes, July)

We find the most similar products to it using clustering (cohorts), then we see the sales historical data, seasonality and forecast the price for the next month. It shows that maybe for these range of products for berlin in summer we will have around M100€ sales.

Then we can see in the whole market (analysing the competitors’ websites) what types of product are usually offered along with a chocolate bar.

Also, we can find users who have the same kind of shopping behavior and try to see what else they usually buy along with chocolate.

And we can do A/B testing. Letting the chocolate bar run for this target market with something else and see how it affects how it is liked. (or a set of products once with Chocolate as a cross-selling object, once without) We can get the data from CDP platforms to find out if the number of clicks have increased or look at the heatamp of clicks to see if the product gets any more clicks or not. Then we use techniques such as Chi-square or t-test to find if the difference is meaningful or not. Then we can recommend if a product exchange will increase the sales. What we need to be aware of, is the diversity of a basket. I think there should be sth like a diversity score that should have a minimum values for each category, or offering to customers. For example: if we believe a chocolate type or brand has not sold previously, and it is the only chocolate item (no substitute) we might still need to keep it there if it hurts the diversity score of the offering.

Example for Amazon BI: they find a cohort of customers who might be interested in a certain feature or product, then they do A/B testing. They split the audience to 50-50 for example, then expose the first half with the feature and the other half without and then see if the difference is significant. If it didnt show us a meaningful result, we try to find another tweak in the cohort. For example sometimes a subset of those people might be interested and not all, then we try to run the experiment again. We think if there is any better way of picking our audience.

Notes from case studies:

When we have a sales dataset, we need to see if a product was on promo, what was the inventory condition. It may happen that a product is on promo but not available in stock?

Perishable=fased shodani

In ecommerce, cross selling happens during the check out process. Cross-selling can alert users to products they didn't previously know you offered, further earning their confidence as the best retailer to satisfy a particular need.

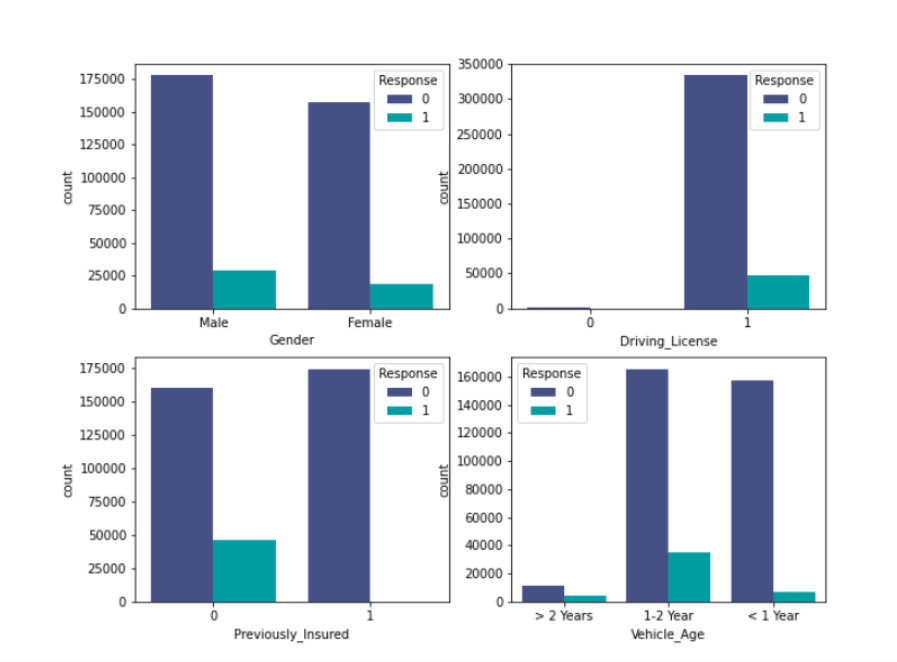
Model Selection: Choose an appropriate predictive modeling technique that suits the cross-selling prediction problem at hand. Commonly used techniques include **logistic regression**, **decision trees**, **random forests**, or more advanced methods like gradient boosting or neural networks.

Monitor and Refine: Continuously monitor the model's performance and make necessary refinements over time.

df.isna() shows all null values in the dataset

df.info() shows the summary of the dataset

Some EDA ideas: find hypothesis, who tends to convert as a (0,1) target feature(people who bought peanut butter, bought jelly too) then count the results and see if the diff between 0 and 1 is meaningful. Example:



Make a histogram of important columns like age to see what age group is mostly responding as yes to the cross selling.

In general, we find some features that are independent to each other and have a meaningful effect on the class prediction or at least we guess it.

Some preprocessing ideas:

* Convert the categorical features into dummies or doing categorical encoding.
* Binning the numerical features.
* dropping the unnecessary columns like ids.

For dummy cols: df=pd.get\_dummies(df,columns=['Gender'] ,prefix='Gender')

For binning: df["Age"] = pd.cut(df['Age'], bins=[0, 29, 35, 50, 100])

df['Age']= df['Age'].cat.codes

If we have a skewed or imbalanced class, we should use resampling to reduce the occurrence of the dominant value to a certain point.

for imbalanced classification problems, the F1 score is a more significant metric

Examples of algorithm hyperparameters are learning rate and batch size as well as mini-batch size.

One of ways to do hyperparameter tuning is using Gridsearch algorithm.

The most effective KPIs are specific, measurable, achievable, relevant and timely (SMART)

SQL: to optimize the SQL speed, join sequence starts from the biggest table. Avoid out join and cross join as much as possible. Always group by the column with the largest number of unique values. Avoid subqueries in where clause.

***REVIEW AND LEARN ABOUT VISUALIZATION AND WHAT VIEW WORKS WELL FOR WHAT KIND OF DATA***

READ STATISTICS BASICS: SOME IMPORTANT ASPECTS:

* Measures of distribution
* Outliers
* Statistical significance
* Regression models
* a/b testing