Introduction

Some recurring themes throughout the answers:

- In case studies, almost never they will explicitly tell you the metric you have to optimize for. Instead, they will just tell you the business goal, such as improve retention, growth, mobile usage, etc. It is your responsibility to mathematically translate the business goal into a metric. If you don't do it, it is an almost automatic fail. If you do it, 90% of the question has been answered
- When they ask you to choose a metric for a new feature, product, or even a department, start from the high level company goal. In most cases, that is (or should be) growth. Then narrow it down. Example:
    - You can grow by increasing user retention or user acquisition
    - You can increase user retention by increasing user engagement
    - I pick metric X cause it is related to user engagement and, if I move it, I can realistically expect to improve company growth
- If you can't eventually link your metric to company growth, that's a huge warning that your metric isn't that useful
- The high majority of metrics need a time threshold in order to be evaluated. Average likes per user can't be evaluated. Average likes per user per day can be evaluated. If the question is specifically about building a metric, don't forget this
- If asked how to improve a product, the easiest way is: focus on actions that you want to incentivize and users are already performing today, but it takes them several steps on the app/website to finish them. There is no better proxy for feature demand that users already doing something despite a complicated user flow. Simplifying the flow will most likely improve your target metrics
- If asked "Should we implement X" or "How to improve Z", the goal is simply explaining step by step how you would find the answer, if you had the data in front of you. They are testing how you approach a problem, not if you are a product visionary
- When asked to optimize a long term metric like retention rate or lifetime value, the question means: find a short term metric that can predict the long term one, and then focus on optimizing that one

- When asked to pick variables, pick a combination of user characteristics (age, sex, country, etc.) and behavioral ones (related to their browsing behavior)

Click here for mock phone interviews to practice similar questions, DS takehome challenges, and more

How would you improve engagement on FB?

Answer:

- Define the metric. Pretty much all case studies start with a vague goal and the first step is translating that into a metric that can be optimized. I.e. "I define engagement on FB as the proportion of users who take at least one action per day where action means interacting with the site, i.e. posting, liking, uploading a picture, etc". As long as it makes sense, it doesn't matter much how the metrics is exactly defined, like which thresholds you choose (once a day in this example) or which actions you choose to include or whether you say average actions per user instead of pct of users above a certain threshold. But it is crucial to define a concrete metric. For instance, response rate on Quora would not be a concrete metric. It can't be evaluated, it is too vague. But percentage of questions that get at least one response within a day is a metric.

  As a general advice, try to pick a metric that is related to the company mission. Like FB cares about interactions between users, that's why above we picked actions that incentivize that. Quora cares about high level content, so you might always want to include some quality measure (ex: responses with at least 3 up-votes within the first day), Airbnb is about "belonging anywhere" so focus on the fact that if you want to go to a given place, you can do it, etc. Especially Silicon Valley companies love these things.

- After the metric, pick the variables you think would matter to move that metric. Almost always a combination of user characteristics (sex, age, country, # of friends, etc.) and related to their browsing/online behavior (device, they came from ads/SEO/direct link, session time, etc.).

- Pick a model to predict metric from point 1 using vars from point 2. Tree-based classifiers are usually your best option. Emphasize why you picked that model. I.e. I pick a Random Forest (RF) cause I want to have high accuracy. A RF works well in high dimension, with categorical variables, and outliers, as I expect to

have here. I will then get model insights via partial dependence and variable importance plots.

- Come up with a couple of possible model output scenarios. I.e. after inspecting my model, I see that users from Argentina are not very engaged. On the other hand, Indians <30 yrs old are very engaged, but proportionally we don't have many of those users. Try to always have one "good" segment and one "bad". Use your knowledge of their product to try to come up with realistic segments. Otherwise, just make them up in a way that are actionable and emphasize that these are just examples, with real data you would find the real segments (after all, the job of a DS is to suggest actions based on the data, not guessing in advance what you will find in the data).

- Define next steps. I.e. I would check the Spanish translation that Argentinians see. Maybe it doesn't feel very Argentinian? We can try with a more localized version? On the other hand, since the site is doing well with young Indians, I would tell the marketing team to reach more of them via ads or specific marketing campaigns. Your case study answer should always end with possible next steps to improve the original metric.

  With minor changes, this template can be used to answer pretty much all typical case study questions (i.e. How do you increase X on site Y).

We are launching a new driver app with a better UI. The goal is increasing driver earnings by increasing their number of trips. Outline a testing strategy to see if the new app is better than the old one

Answer:

The naive answer here would be: I pick a few markets that are representative of the entire population and, on each market, I randomly split drivers in test and control. Then I do a statistical test on the target metric and check if the new app is winning. The reason this would fail is that test and control wouldn't be independent.

Let's say I take all drivers in San Francisco and give 50% of them the new app and 50% keep using the old app. If the new app is effectively making drivers take more trips, that will result in higher competition for the drivers using the old app, and, therefore, will affect their earnings. The opposite is also true. If the new app sucks and those drivers drive less, this will also affect old app driver earnings given that there is less competition. It is extremely hard to design an A/B test in marketplaces or social networks since users are all connected.

If you get a question like that, to quickly check if you can simply randomly split users, mentally take extreme cases. Let's say the new product has a bug and it is unusable. Or the new product is amazing and test users will use it 24/7. Will these options have any effect on the control group? If the answer is yes (like it is obvious for the Uber case), you can't just randomly split users.

Best way to answer:

- State why you can't randomly split users: this is really what the question is looking for.

- Therefore, decide you will test by market. That is, you will match comparable markets in pairs and, for each pair, give everyone in one market the new app and in the other one the old app. Comparable means that, during the time of the test, the main metrics are expected to be very similar, if there were no test.

- To choose how many markets you need, define in advance sample size needed for a t-test. To identify required sample size, choose power, significance level, minimum difference between test and control, and std deviation. Review how to estimate t-test sample size. It is often asked.

- Run the test and, after having reached the required sample size, check if results are significant.

- [Bonus]: check for novelty effect. Users use more a product when it is new. Not because it is better, but simply because it is new. As novelty ends, they will use it less. This is called novelty effect and often makes tests look like winners when they are not.
  You can control for this by, in your results, subsetting by drivers for which it is the first experience. Novelty effect obviously doesn't affect new users. If a test is winning overall, but it is not winning when comparing new users in test vs new users in control, it is a big warning that there might be novelty effect [more on novelty effect in other questions in this ebook since it is a common topic, often asked like this: "We ran a test. It won by 5%, but, after making the change for all users and waiting for a couple of weeks, we didn't see any improvement in our metric. Why?"].

Give an example of a case in which you run an A/B test, test wins with significant p-value, but you still choose to not make the change

Answer:

There are many variations of this question. Sometimes they actually show you some test results with very little impact on the target metric and ask you what you would do. Sometimes they just directly ask: "what are the costs of making a change". Or, like in this case, they start from the conclusion (not making a change) and ask you to explain why. Those are essentially all the same question.

Making a change on the site has some costs by itself:

- Human labor costs, such as engineering time to make the change, product manager time to document it, customer service time in case some users get confused and reach out to customer service representatives with questions. Another way to look at this is opportunity-cost. I.e. all these people are not working on something else with a possibly higher expected value.
- Risk of bugs. Whenever you touch the code base, there is a non-zero chance that something will go wrong.

If you are adding complexity, you might also be increasing the costs above for future changes. That is, you are increasing the complexity of the code base and, therefore, in future there will be higher risk of bugs when you make a change or it will require longer engineering time to make a change. So your cost estimation shouldn't only be short term, but should also include long term considerations.

As a general rule, simple is much better than complex. So in order to justify a change that adds complexity, the gain should be relevant. The actual definition of "relevant" is typically not a data scientist responsibility. Product

managers or product executives will typically tell you what's the minimum gain they are looking for.

If you have a good understanding of inferential statistics, this is also a good opportunity to show it. When sample size is very large, as it can be the case with tech company A/B tests, very small differences will become detectable. So, with large sample size, it is extremely likely to get a significant p-value, even if the effect is very small. In those cases, it becomes much more important to look at the effect size than the p-value to decide if it is worth going ahead with the change.

Do you expect that Uber trips without rider review have been better, worse, or same as trips with reviews?

Answer:

This question is about missing values in a very specific situation. If a rider is not leaving a review, you are going to have a missing value in the "review" column. The goal here is to guess rider experience without the rider explicitly telling us.

Since it was a rider choice to not leave a review, we can pretty safely rule out the option that missing values follow the same distribution as non-missing values. So you can't just assume that missing values will have the same average or median of that variable. Please, note that the high high majority of missing values are not random and, in most cases, are affected by self-selection bias. For instance, not filling out profile information on a social network is another clear example of non-random missing values.

At this point, guessing missing values becomes a standard machine learning problem. You have a bunch of labeled data (rider giving a review) and you need to predict unlabeled data (missing values). So you can build a model to predict reviews using variables related to the trip (such as waiting time, trip duration, cost, driver and rider information, time of the day, destination, etc.).
Another way to look at this is: for the trips without a review, try to find trips with reviews with similar characteristics and, based on that, predict the review. At the end of the day, once you remove the hype, this is what machine learning does.

In practice, it is realistic to expect that the majority of trips without a review will be related to relatively bad trip characteristics, i.e. long waiting time, delayed arrival time, bad car conditions, a driver with bad reviews, etc. People, especially from certain countries, feel uncomfortable giving a bad review and prefer not giving it at all, if they had a bad experience. And, by the way, companies keep running experiments to try to reduce this issue and incentivize users to leave a review no matter what.

Finally, here the question is simply about the distribution of missing reviews. So it makes sense to try to guess their value via machine learning. However, if you were using reviews as an input variable to predict something else, you would be better off not replacing missing values and simply recoding them as -1 or something like that. There is a lot of information in the fact that the user has chosen to not do something and you don't want to lose that information by replacing missing values with some guesses.

A given category of an e-commerce marketplace, for instance *jeans*, is not doing well. How would you estimate if it is a demand or supply problem?

Answer:

Demand and supply in marketplaces are so strictly related that it is often really hard to understand whether you should focus on improving supply or demand. Demand often follows high quality supply and, in turn, supply follows large demand.

Considering a typical site funnel (i.e. home page, search results, click on item, buy it), going to a site and searching for "jeans" can be seen as a proxy for demand. Those are users with at least some intent to buy jeans. The next steps, clicking on a given search result and converting, can be seen as a proxy for supply.

A good proxy for demand can be ads click-through-rate (CTR). Run an ad campaign specifically about jeans. If campaign CTR is high, it shows demand should be there. In practice, that means you will have many users coming to your site looking for jeans.

A good proxy for supply can be # conversions/# searches. So your denominator here are already people with some sort of intent to buy jeans (i.e. they searched for it). If the site fails to make them convert, it is often because supply quality or quantity is not good.

A way to refine the supply metric would be to better define people with intent. The more specific that definition is, the more accurate the metric will be. For instance, you can start by only considering people who used filters in their search. Or people whose session time is above a given value. This will remove some noise and only give you users with really high intent. If these people are not converting, there is a high chance your supply needs improvement.

If there is a supply problem, filter usage is also a good way to understand where the problem lies. For instance, high usage of price filters with very low conversion for those users likely means that your supply is too expensive.

Click here for mock phone interviews to practice similar questions, DS takehome challenges, and more

What are the drawbacks of using supervised machine learning to predict frauds?

Supervised machine learning here means: you have a training set with fraudulent activities (label 1) and non-fraudulent activities (label 0). Train your model and use that model to predict future frauds.

Answer:

This is the same as asking what are the drawbacks of supervised machine learning in general and, especially, when you have extremely low signal-to-noise ratio.

The high majority of events will be legitimate (thankfully). Therefore, your model will be happy predicting everything as non-fraud, achieving very high classification power. But this is not particularly useful. There are ways to fix this, such as changing the model internal loss to penalize more false negatives, using an extremely aggressive cut-off point for classification (i.e. everything above 0.1 is classified as fraud), or reweighing the training events. However, these techniques only work if you have many positive cases in absolute numbers in your training set. And to have many positive cases you need massive amounts of data. This can be true for large companies, but not for small companies.

Also, your training set is based on past events that have been identified as frauds. If people have committed frauds in the past, but you haven't caught them, not only your model won't be able to predict them as fraud, but it will actually be trained to recognize those as legitimate transactions. This can create a very dangerous

vicious circle where more and more of those frauds will happen.

Finally, and perhaps most importantly, this approach provides no defense towards people who figure out new ways to commit frauds that don't follow previous fraud patterns. You will always be one step behind:

- People find new ways to commit a fraud and manage to steal money
- At some point, you realize that and retrain your model with the new positive cases
- By that time, a new fraudulent technique has been developed

A more reliable technique to prevent frauds is using anomaly detection. In this case, you define as fraud any transaction having a pattern which is significantly different from legitimate transactions. That theoretically solves most of the problems above, since you are not training your model on specific kinds of frauds, but simply on different behavior. However, it suffers from its own limitations. For instance, in high dimension, you are almost guaranteed that every transaction is an outlier on at least one dimension (curse of dimensionality), so the anomaly detection approach requires a significant investment in terms of time to exactly define what means being an anomaly.
As often, a combination of both approaches, supervised machine learning and anomaly detection, to minimize each other weaknesses tends to be the best thing: block users if their pattern is similar to past frauds or if they are anomalies.

If 70% of Facebook users on iOS use Instagram, but only 35% of Facebook users on Android use Instagram, how would you investigate the discrepancy?

Answer:

This is really just another way to ask: we are trying to predict Y (here Instagram usage), how to find out whether X (here mobile Operative System - OS) is a discriminant variable or not. And follow up question is: if not, what are the actual discriminant variables? If yes, what are the actual differences between the operative systems that might drive the outcome?

So, let's just go step by step. Is OS an important variable in trying to predict Instagram usage? A pretty likely guess is that OS is just a proxy for other variables, such as age, country, education, engagement, etc. It is not OS that's driving Instagram usage, but, for instance, US-based people are more likely to have iOS and US-based people are more likely to use Instagram.

A quick way to check this could be: build a decision tree using user-related variables + OS to predict Instagram usage. If the tree doesn't significantly use the OS variable, you are sure that it doesn't really matter. In this case, the tree would split on, for instance, combinations of country and age since that's where most of the information is. If OS is acting as a proxy for those variables, for sure the tree would choose those other variables first, as they would allow for a better classification. After those splits, there is no information left to extract from OS so the tree won't touch it. It is important that you choose a model that works well with correlated variables, like trees.

You could get similar conclusions by building two models: one including OS and one without. If prediction is the same, it means there is no information in OS that's not already in the other variables.

Another option would be to generate simulated datasets where you adjust the distributions of all other variables. So that now you have the same age, country, etc, distribution for both iOS and Android. And check if proportion of Instagram users is still different or not.

If you found out from above that OS is just acting as a proxy for other variables, you are done. Just look at the top tree splits and see the real reason of the difference.
In the (unlikely) case that you found out that the problem is actually OS, you need to investigate the reason. At this point, the most likely guess is that one app is much better than the other one. So collect all variables about the app (such as loading time, user clicks, bug reports) and see if you can find any significant difference between the two apps. Again, you can build a model using those variables to separate iOS vs Android. You might find out that loading time is significantly worse for Android, or people from the home page are not clicking where they should (bad UI), or there were more bugs in the past and this has affected long term growth (people stopped using it after trying the buggy versions and never came back).

We made a change to our subscription offering adding new features. We expect this to increase subscription retention. How can we test if the change is successful?

Answer:

This question goes under the category of "how to test something that takes a long time to be evaluated". You can't run an A/B test for months. So, whenever your metric needs a long time to be evaluated, you need to find a short term proxy for the long term metric. This is the point of this question as well as similar questions about long term user engagement, retaining users, or estimating customer lifetime value.

Firstly, as usual, let's define subscription retention. A common metric is percentage of users who don't unsubscribe within the first 12 months. As should be clear, we would need to run the test for more than 12 months to make sure the test is successful, which is unrealistic.

So this question really becomes: which short term metrics can be used to estimate subscription retention? A common approach is:

- Collect subscription retention data from people who subscribed more than 1 year back and label each user as 1 if was still subscribed after 12 months and 0 otherwise.
- Collect variables related to user behavior and segment them by time. For instance, start by picking as time limit one week and pick variables about user behavior within the first week since she subscribed (i.e. did she unsubscribe? Did she use any of the subscription products? How many times? etc.).
- See if using any of these variables you can accurately predict long term retention. This could be just one

variable, for instance you found out that proportion of people who unsubscribed or never used my product within the first week is a perfect proxy for long term retention. Or could be done via machine learning. You build a model to predict long term retention with those variables. In this case, the model output becomes your proxy for long term retention.

- You can now run a test where your target metric is based on the point above and can be evaluated after just one week. If you manage to increase the short term metric, you assume you are also increasing the long term one. After all, you proved that the short term metric can predict the long term one (obviously this assumes that the new subscription features don't dramatically change user behavior, otherwise your model would not be useful anymore).
- If you can't predict long term retention after one week only of data, try two weeks, then three, etc. Very roughly speaking, three weeks of data should be enough to predict long term behavior in most cases.

Do you think it is better to target ads based on user demographic or behavioral characteristics (past browsing experience)? Why?

Answer:

A similar super common question from Google is: tell me all variables that can be useful in predicting ad clicks and, among those, tell me which ones are the most important. It is really the same kind of question.

This is a common question for two reasons: as a data scientist, you might have to build a model to predict ad clicks, like in the Google example above. So it is important to get a sense of which variables would matter and why. But, perhaps even more importantly, as a data scientist your task might be to help build a product whose goal is to collect data that can improve ad clicks. And, therefore, you need to have a strong sense of which data will be useful to then build a product that can serve your purpose.

Your goal here is to have data to predict probability of clicking on an ad. There are two key differences between demographic and behavioral/browsing data.

- People are actually really likely to buy products not only for themselves. It can be a gift, it can be a relative or friend asking them to buy it on-line for them, or many other reasons. Models only built on demographic data have no information about this, but assume users will buy for themselves. Looking at a user browsing history gives information about what a user is interested in buying regardless of whether it is a gift or for herself.

- In advertising, timing is the most important thing. You want to predict when a user is interested in buying something. Demographic information can tell you, for instance, that users of a certain age tend to buy a certain product. Browsing data tells the moment in which a certain user is thinking about buying a product. This is orders of magnitude more useful, especially in the online ad business that's largely based on monetizing clicks. So they make money if the user clicks on the ad right when they show it to them.

Famously, Facebook revenue spiked when they moved from demographic-based ads to using browsing data.

Always try to keep this present also when they ask you questions about implementing new features. If the company makes money via ads, think about which variables can increase ad clicks and see if the new product/feature can positively affect them.

We ran an A/B test. Test won, so we make the change on the site for all users. But after waiting for some time, we realize that the new version of the site is not performing better than the old one. What could be the reason?

Assume there was no technical or statistical problem with the test, and the reason has to do with user behavior

Answer:

This question is also really common and can be rephrased in tons of ways. Sometimes they directly ask you about novelty effect, but more often they give you some test results and you have to explain them.

The main reason a situation like the one described in the question happens is because of novelty effect. That is, when a site puts up a small new feature, at first users tend to engage with it a lot out of curiosity and to try it out. As novelty wears off and users get used to it, they go back to their normal behavior.

Obviously, this can have a very bad impact on test result reliability, since tests measure the impact of a change over a small period of time and use that data to estimate what would happen in the long run.

As a data scientist, you should always try to use data to validate your guesses. Just saying "you think it could be novelty effect" won't be particularly useful. A good proxy to check for novelty effect is:

- From your test results, only consider users for which it was the first experience on the site. First time users are obviously not affected by novelty effect.
- Look at test results between new users in the control group vs new users in test group
- If test group is winning overall, but not winning for the new user subset, it is a big warning that your results might be affected by novelty effect. Especially in cases like the one of the question where you already know that some time after making the change, you saw no improvement.

Comparing new user results vs old users, in cases where you strongly suspect there is novelty effect, can also be useful to get an actual estimate of the impact of novelty effect. And this can be used to control for novelty effect gains in future tests.

You are launching a messaging app. Define 2 metrics that you'd choose to monitor app performance during the first months. Why you chose them?

Answer:

The answer below is not just for messaging apps, but applies to pretty much any new product launch. Ultimately, you want as many users as possible using your product and you want the number of engaged users going up over time. An old business adage says that your product is either growing or dying. And you don't want it to be dying.

When they ask you for a metric, always start by targeting growth as the high level goal, and then narrow it down to something more specific that should have an impact on growth. You typically need two components in order to have product growth:

- Ability to acquire new users
- Ability to retain current users

It is really hard to grow if you lose your past users and, certainly, not sustainable in the long run. It is pretty much impossible to grow if you don't increase the user base. Therefore, you should pick two metrics that reflect the two bullet points above.

Regarding acquiring new users, a good metric could be: new sign ups per day from users who send at least 1 message within the first 2 days (the at least 1 message within X day part is such that you have some measure

of quality of sign-ups).
Regarding engagement, pick the key action you want your users to perform. If possible, pick actions that incentivize other user actions. That will create a virtuous circle. For instance, here, we can choose average messages per day per user.

Avoid vanity metrics, i.e. metrics that look good, but are useless. A good way to quickly identify if my metrics are useful is: imagine a bot is creating a bunch of fake accounts. Will my metrics go up? In the example above, we are good. The engagement metric will drop and alert us that something not good is going on. On the other hand, if you ONLY looked at new sign ups per day, you would be happy with the fake accounts. If they ask you for just one metric, make sure it is not a metric that a bot could improve.

Finally, please note that engagement and retention are often two sides of the same coin. In the high majority of cases, retained users are the most engaged ones. So, if they ask you about retention, start from defining engagement.

Which feature would you add to WhatsApp? Hypothetically, imagine you have access to all WhatsApp-related data, such as messages and where people click on the app.

Answer:

You are almost guaranteed to get a question similar to this during the recruiting process. It could be: "How would you improve my product" or "Pick your favorite app, and tell me how you would improve it" or "Why do you think we added that feature".

You might be tempted to answer this as if you were a product visionary, i.e. "I think we should add this because I think people really want it and that's where technology is going". But, really, avoid these kinds of answers. They are hiring you as a data scientist, as someone who looks at the data and, based on that, suggests product improvements.

The easiest way for a data scientist to find product opportunities is by looking at current data and find something that people are already doing, but in a complicated way requiring multiple steps. Then, simplify that. Data science excels in identifying and removing inefficiencies. Whenever you get questions about new features, answer from that standpoint. It is the safest bet.

Let's consider WhatsApp. Many of the things you can do on WhatsApp with one or even zero clicks can be seen as having simplified something that in the past would require a few steps or actually typing a message

(clicking is much easier than typing). Btw by telling you "you can hypothetically go through user messages", they are already hinting at what they want to hear.

Some examples of simplifying:

- Sender often asking the other person if they had received/read the message ("are you there?" "did you get it?", etc.) -> put checkmarks to let users know directly
- People often writing messages about scheduling a phone call (i.e. "let's talk by phone", "can I call you now", etc.) -> add voice call within the app
- Live location or constantly updated time to destination from within the app. It is likely fairly common for people to send messages like: "where are you", "sorry running late", "almost there", "stuck in traffic, there in 15 minutes", or even "text me when you are home". That feature allows to let the other person know all this with just one click
- Sending the same message to multiple people -> broadcast feature

A step by step approach to identify these opportunities could be:

- Collect messages and cluster them based on their meaning. Focus especially on the first and last of a conversation, as they tend to be the most informative ones
- Look at the most frequent clusters
- Figure out a way to allow a user to perform that action with just one click (or even zero) without having to type a message and/or without leaving WhatsApp
- Test on a subset of users if the change is actually successful

An example of this could be identifying that the last message of a conversation is about calling Uber, ordering food, or using any kind of other app. And a possible next step could be to integrate that functionality from within WhatsApp, kind of like Google Map can be called from inside WhatsApp.

Just to emphasize again, when you answer these questions, the most important thing is describing the approach you would take. The question you have to answer is: "How would you find out product opportunities", not what you will actually find out. The second thing depends on the data and you have no access to it yet. The first one shows how you approach a problem, which is what they care about.

You have to predict conversion rate on Airbnb using user country as one of the input variables. How would you deal with the missing values in "country"?

Country is missing in the dataset if the user chooses to not select her country when she creates her profile after signing up

Answer:

A pretty standard question on how to deal with missing values in your input variables when you have to build a model.

There are many ways to deal with missing values if you go by the books. The most cited ones are to replace them with the median/average or build a model to predict them and then use the predicted values.

In practice, these things tend to make more sense in a book than in real life. Firstly, because the high high majority of missing values in tech companies come from the fact that the user has chosen to not give that kind of information: by not filling out her profile, or by changing her privacy settings, or even by deleting her cookies. And that's crucial information for your predictive model that you would lose, if you were replacing the missing values.

Secondly, if you could predict the missing values using other variables, then just include also those other

variables and use a model that works well with correlated variables, like tree-based models for instance. Using the other variables your model would still be able to extract country-related informations + it could extract information from the fact that the user chose to not provide that information.

The safest approach is therefore to use some label like "no_country" when the value is missing. And if by using other variables you can predict the actual country, for instance using a combination of ip address and service provider, then include those variables too in your training set. So for each user you can know both where she is based and whether she chose to tell you her country.

How to estimate the value of a user coming to your e-commerce store when they land on your home-page for the first time?

Answer:

This question is often also rephrased as: "How much should we pay for a click on our ad that leads to our site". The two questions are exactly the same. And, matter of fact, this is also pretty much the same as asking to estimate a user lifetime value (LTV).

Firstly, let's define the value of a user as total revenue coming from that user within the first year since the first time she comes to the site. In sites with network effects or where users are particularly likely to bring additional users to the site, the actual value of a user is higher than that. But for an e-commerce site our definition is reasonable. Most importantly, in a job interview you want to be as clear as possible. Always reduce the complexity of a problem and then solve it. Much better than keeping adding complexity and corner cases and, eventually, not arriving at any answer because you have too many things to take it into account.

Not differently from the subscription retention problem then, the challenge here is that you have to predict something that takes a long time to age (1 year per our metric definition) using short term data as features (only up to the point they land on the home-page for the first time).

Let's collect our training set. Let's go back in time and pick users who came to our site more than 1 year back. We do know the revenue coming from them.

Let's then collect the features. As usual, browsing behavior prior to coming to my site. For instance, did they

come via SEO/ads/direct link. If SEO, what did they search for? If ads, what was the ad text? Came from Google/Bing/Yahoo? Also, important features are user location, user device, operative system, type of browser as well as browser locale as a proxy for language. Week of the year can also be very relevant.

Once we have this data set, we can use those features to predict revenue per user and we are done. Any model that can predict a continuous output will work here. Regression trees can be a good choice since many of the variables we have are categorical + the most likely output of a model like this is to identify a few segments, such as US/UK users using iPhone, European users from ads, etc., and assign a predicted $ value to each segment. So trees are perfect.

How can we tell if two users on Facebook are best friends?

Answer:

This question is just a fancy way to ask: "Given an item (i.e. you) and a list of item candidates (i.e. your connections), find the most similar item". If the question were "For each song on Spotify, find the closest song", it would be very much the same.

As usual, let's start by defining what means closest or most similar in this case. There are infinite ways to define best friends. Pick one definition that makes sense and move on. What they really want to see here is that you define it in some ways. If, by the end of this answer, you spent all the time talking about all possible definitions of best friends and didn't talk about item recommendation from a data science perspective, you know you have failed.

So let's say best friends are those who spent the majority of free time off-line together. This can include your partner, a relative, or really anyone else as long as you choose to spend the majority of time with this person.

So now let's pick variables that can be seen as a proxy for closeness as defined above. For simplicity, we are not adding a time factor here and just look at the entire user history without giving different weights based on time recency.

Whenever you have networks and want to define the strength of interaction between two items, look at the overlap between each item network. That is, if person A and person B share many connections in common, that's a pretty strong sign that A and B are closely related. Matter of fact, this is also not that different from the

idea behind google search ranking to match search queries with the closest results. And, in machine learning, if you want to see how close are variable X and Y, a very reliable approach it to check if they interact a lot with the same group of variables.

A more refined approach here is to use not only the absolute number of shared connection, but to also add if two people interact with different clusters of people. Let's say I cluster connections into groups (for instance, work friends, high school friends, university friends, etc). If two people share connections across different groups, it is a very strong indicator of closeness. In the real world, it means that person A is introducing to person B her co-workers, university friends, etc. and the other way round.

Other proxies for closeness could be pictures where they show up together. Also, events they attended together via Facebook event page. Number of comments on each other pictures/wall posts can also be useful. Even more useful would be scraping all comments and look for anomalies. The high majority of FB comments are from acquaintances and follow a pretty standard template. Best friends are likely less formal and write in a different way to each other.

Once you have all the variables above, you can use any similarity measure (like cosine similarity for instance) and, for each person, find the closest connection.

Which variables are important to predict a fake listing on eBay?

Answer:

Some general advice when dealing with fraud:

- People hardly commit just one fraud. They do as many as possible, once they find a successful strategy. However, a user, in order to commit multiple frauds on a site, needs to keep changing user_id and create new accounts. Therefore, all those variables that should be unique by ID can be very useful to identify fraudulent behavior. This is because they allow to track the same user across different profiles. For instance: device ID, IP address, physical address, credit card number, etc.
- No matter what, the high majority of a given site users won't commit frauds. Therefore, looking at variable distributions and finding outliers, often called anomalies in fraud detection, is a very good strategy to identify suspect behavior.

As you will see in the examples below, all variables are really about something that should be unique and is not or extreme values.

More specifically about eBay, when you have to predict frauds in a peer-to-peer marketplace, break down your answer into two categories: characteristics of the listing (or, more generally, of whatever is sold) and characteristics of the seller.

Regarding the **listing**, some variable examples are:

- Pictures characteristics like very low resolution or quality. Especially important is a variable that tells whether that picture is original or is taken from the internet (you can verify this via an image search and

calculate image similarity)
- Price, especially if too low
- Words used in the description, especially if exactly the same description can be found on other listings

Regarding the **seller**, some variable examples are:

- Device ID, IP address, bank/Paypal account, any personal information provided during the sign-up flow. Trying to see if any of these can be matched to other sellers on the site.
- Ratings, especially if they have no ratings yet and the account is brand new
- Browsing behavior that led to the seller creating the account. This is really important. A new seller is expected to spend a significant amount of time on the site, before and right after creating an account, to learn how it works. A new seller that right away creates an account and, from her behavior, looks like is already familiar with the site is very suspect.

Explain the drawbacks of running an A/B test by market (i.e. all people in one market get version A of the site and another market version B)

Answer:

Testing by market is extremely common. In many cases, it is really hard to split users into independent groups, especially if your site is a social network or a marketplace where actions of one user have an effect on other users.

A very common approach in those cases is testing by market. Find markets which are similar and whose users are not connected. Then match them in pairs and give users in one group of markets the test version and users in the other group the control version.

Some examples of drawbacks of this approach are:

- Tests by markets are more noisy. Simply cause users in different markets will never be as similar as users in the same market
- Assuming full independence between users in different markets might be a stretch in certain cases
- Even if past metrics show similar behavior (and that's part of how you match markets), this might not guarantee that users will react the same to a new feature
- Consider a pair of markets. Something unpredictable might happen during the test just in one of the two markets. For instance, a competitor launches an aggressive marketing campaign there, political events, etc.

Some good practices when it comes to testing by market are:

- Check one metric that's not supposed to be affected by your test. Make sure that during the test keeps behaving similarly for both markets
- Only test relevant features, i.e. features that are supposed to have a large impact on user behavior. Tests by market are more noisy and, therefore, you want to test something that, if successful, is expected to deliver gains well above the noise

How would you measure the performance of the customer service department?

Answer:

The most successful companies have growth as a goal for pretty much any dept. So when asked about metrics for given departments or teams, it is a safe thing to start by saying: the goal should be increasing company growth. It is really hard to justify employee costs if their work can't concretely be linked to growth.

Growth can be achieved in two ways: getting new customers or retaining old ones. No matter in which dept. someone works, she should be able to show that her work is either helping in getting new customers or retaining old ones. To sum up this in one metric, customer service should increase average user lifetime value (LTV). If the user didn't have any transaction in the past, this can be achieved by helping through the first experience. If the user is an old customer, this can be achieved by making sure that the user has a as positive as possible post-sales experience.
Note that, since the cost of acquiring a new customer is in general really high, the ability of customer service to retain users is worth a lot of money in the long run (which is also why they tend to give coupons easily to make the customer happier if something went wrong. Still cheaper than having to acquire a new customer).

Now the problem becomes a standard LTV calculation problem. As usual for long term metrics, the first step is to find a short term proxy for the long term metric. Otherwise, if we define LTV time window as 1 yr, we would need to wait for 1 yr before knowing if customer service is working well!

So:

- Pull data from more than 1 yr ago and take all customer service tickets

- To make it simpler, let's make it a binary problem and define tickets as 1 if the corresponding user bought within 1 year after the ticket, 0 otherwise
- Use NLP to extract information from the user-customer service text interaction (like sentiment analysis and topics). And add things like response time and user feedback right after the ticket.
- Build a model to predict the binary variable
- If you find that one single variable is highly predictive of the long term metric, just use that one. Often user feedback will work. But you must have proved it via the steps above that you are using that variable because is predictive of the long term growth-related metric you care about.
- If you can't find any single variable, but you need several variables to predict LTV, you can simply use the model output as the metric to optimize

FB - Should we add a love button?

Answer:

This is another super common question type, where the candidate is asked whether it would be a good idea to implement a new feature. Although it is not obvious from the way the question is phrased, even questions like this one should be answered following a very quantitative framework.

This is a particularly important question since a big part of the job of a DS is to look at the data and, based on findings, suggest new feature/product ideas.

Break down the answer into two steps. Firstly,

- If the feature were highly successful, would it be a good thing for the site?

To answer this, pick a crucial metric for the site that that feature is supposed to move. If you can't find any, then there is no point in even discussing further.
In this case, we pick engagement as the key metric, where engagement is as usual defined as number of actions (likes/posts/comments/love/pics uploads/etc) per user within a certain time frame.

We can realistically expect that if the feature becomes highly popular, that will benefit engagement for two reasons: clicking on the love button itself will improve engagement + it might incentivize posting new statuses. After all, receiving a lot of "love", or positive attention in general, is a nice feeling and will make people more likely to post.

The real key part of the answer is the second part though:

- You need to find a proxy for demand of that feature in your current data

The safest way for a data scientist to drive new feature decisions is to look at what users are doing today on the site. See if you want to incentivize that behavior and, if so, try to simplify it, so that users can complete it

more easily. The fact that users are already doing it today shows demand. Since there is demand, simplifying it will for sure increase the number of users doing it.

For instance, if many users are performing some sort of activity on your site and that requires several clicks in a row, find a way so that they can just do it with one click (think about buying with one click for instance). Please, note that the converse is also true. If you want to disincentivize some actions, add complexity (think about how many clicks it takes logging out or contacting customer service on many sites).

So, back to the Facebook example, we need some data that can be used as a proxy for demand for that feature. The easiest thing to do is to look at current comments and use NLP to extract sentiments. Are there many comments that can be broadly assigned to a "love category"? If so, that's a pretty strong indicator of demand for a love button. Indeed, that's telling us that today users are going through a longer path to express love (i.e. actually typing a comment instead of just clicking on a button) and it is realistic to assume that, as you simplify the path, you will have even more users doing it, i.e. all those people who share the same feeling, but don't feel like going through the multi-click path.

Please, note that even if your conclusions are that (a) the feature, if successful, would benefit the main metric, (b) you found a proxy for large demand, and (c) you are simplifying the path, this doesn't mean you should implement the feature. It means you should test it. Then, if and only if the test is successful, you should implement it. There is no way to know if that feature will actually move an important metric without running an experiment.

How would you use data to evaluate if it makes sense to implement two-step authentication when users log in?

Answer:

This is a very common question in fraud. Sometimes is phrased like above, sometimes like "Explain how you would choose the right threshold in a machine learning fraud model". Sometimes they ask you to talk about the trade-off between false positive and false negative in a fraud model. Sometimes they directly show you a ROC curve and ask you to pick the best cut-off point. They are all the same thing.

Let's say you have to build a machine learning model to predict fraudsters. You will then pick a threshold. That is, if model output is >X, you flag the user as fraudster and block her. Otherwise, the user will normally go through the site. The lower is the threshold X, the more true positives you will catch, but also false positives. Meaning you will save some money as you block people who wanted to commit a fraud, but you will also lose some money since you block legitimate users.

Once you define the cost of a false negative (i.e. what's the cost of a fraud?) and the cost of a false positive (blocking a legitimate customer), you can then optimize the threshold accordingly.

Two step authentication is exactly the same thing. It has the same consequences as increasing the threshold in your model. You will lose some legitimate users since they won't bother to go through the two-step process. But you will block a lot fraudulent activities.

To sum up, a first approach would be:

- Identify cost of false negatives (actual frauds happening) and value of true negatives (value of a legitimate

user) from a business standpoint by looking at past data

- Run a test where only a subset of users is required to go through the two-step process
- Apply values from point 1 to results from point 2 and see if your test is winning. That is, is the number of bad actors that two-step is blocking worth the number of good actors that the site is losing since it is harder to log-in?

The approach above will lead to potential improvements overall, if the test wins. But it is hardly the most efficient strategy. If I have a model that tells me that a user has 0.01 probability of committing a fraud, why should she go through the two-step process? If you only look at users with probability of fraud < 0.1, the two-step test is probably losing money within that segment no matter what.

A more common approach is to segment users according to predicted probability of fraud. And then finding for which segments is profitable to implement the two-step process.

And this is exactly what happens in practice. If you try to log-in on FB or Gmail, it is usually a simple one-step process. But if some model predicts that it might be a fraud (for instance if you log-in from a different country with a different device ID), it will ask for two-step authentication.

At Facebook we use as a metric number of likes per user per week. And, each week, we check it year over year to control for seasonality. This week the metric is dramatically down. How would you find out the reason? Logging is fine as well as the query we used to get the data

Answer:

Whenever they ask about why a metric is down, start by breaking it down into its components and analyze them independently. Very often in these questions they give you year over year metrics (i.e. this year divided by previous year) and the metric itself is a proportion. The goal of this from their perspective is to see how you, step by step, break it down into numerator and denominator.

- The first step is where the majority of people fail. They assume year over year down means this year is down. Not true. Year over year down can be this year down (numerator down) or last year up (denominator up). Now make an assumption. Say: let's assume I found out that numerator is down.
- So now the problem could be number of users suddenly up and these additional users are not liking as much as the usual ones. Or number of users is normal and number of likes suddenly down. Let's assume there was a huge spike of users and likes are constant, meaning the new users are less engaged.
- So now we need to figure out where these new users come from, why they are less engaged. A fast way could be: take all your "up week" users and label them as 1. Then take all previous week users and label

them as 0. Take a bunch of variables related to the users. As usual, behavioral and demographics related variables. Now build a binary model to separate the two groups. A decision tree is perfect here.

- Look at where the tree splits to create a leaf with higher proportion of 1s. Those splits will tell you the characteristics of the additional new users. For instance, you could have way more users from China this week. This might depend on a marketing campaign there that got a huge number of users, but these users are less engaged, as often when users come from sudden marketing campaigns. Or you could see that all these new users come from very few different IP addresses. That would mean that all these users are probably fake accounts.

As an advice, as you break down the metric and make assumptions along the way, try to end up in a scenario where nothing actually happened on the site, but it simply changed the user distribution. It makes much easier to give possible reasons as the last step. You can simply take a random country and say users increased there for a local marketing campaign or decreased because of a local competitor launch. It just makes easier to clearly define and delimitate the problem.

But you could certainly also say that actually number of likes went down and then explain how you would discover the reason. If a number related to an action on the site suddenly goes down, the most likely reason is a bug.

We are running 30 tests at the same time, trying different versions of our home page. In only one case test wins against old home page. P-value is 0.04. Would you make the change?

Answer:

This question is about your understanding of hypothesis testing and, especially, what happens when multiple hypotheses are tested at the same time.

In a statistical test, you reject the null hypothesis when you observe unlikely events, if the null hypothesis were in fact true. Said it in another way and focusing on a standard A/B testing framework: test wins against control, and therefore you make the change on the site, if test is better and the results are unlikely, under the assumption that there were no difference between test and control.

The problem is that, if you are running multiple tests at the same time, you are increasing the chances of finding unlikely events by probability only. Imagine you flip a biased coin with 5% probability of getting head. If you do it long enough, you will get a head. For the same reason, your significance level threshold in the tests needs to be adjusted to balance the fact that more experiments are increasing the probability of finding unlikely events.

In practice, the most common approach is to use the Bonferroni correction, simply dividing 0.05 by the number of tests and this becomes the new threshold for significance. Another common approach is using false

discovery rate.

The answer is therefore no, I wouldn't make the change. Indeed, if I were using Bonferroni, I would make the change only if test is better and p-value were less than 0.05/30.

Please, note that in this case the question was very straightforward. But you might also get it in a more subliminal way. For instance, they show you one test results. Test is not significantly different. Then, they segment the results by say 20 countries with large enough sample size. And show that one of these countries is actually significantly better with 0.04 p-value. Then they ask you what you would do. This case is exactly the same as the original question. By looking into several different segments, you are increasing the probability of finding at least one winner by chance. And, therefore, you should change the significance threshold accordingly. In this example, if you only looked at the 20 country segments, you could make it 0.05/20 using Bonferroni.

Each user on our site can be described by 100 continuous variables. What's the probability that a user is an outlier on at least one variable? What are the implications of this from a product standpoint?

Define outlier as being in either the top or bottom 2.5% of each variable distribution, meaning there is a 5% chance to be an outlier on each variable taken independently. All variables are independent from each other.

Answer:

This question has two parts: the first is solving a probability problem. The second one is understanding why this is important when you have to build a product.

For the probability part, the probability of being an outlier on at least one dimension is 1 minus the probability of being within the 95% range for all the variables. That is, 1 - 0.95^100. This number is larger than 99%!

That essentially means that in high dimension every event is almost guaranteed to be an outlier on at least one dimension. If you imagine an n-dimensional space, where n is the number of features, all events will end up close to the border and the center will be empty. This is pretty much another way to rephrase the curse of dimensionality and has huge negative implications when it comes to modeling.

The implications of this phenomenon are also product-related and at the very core of why companies need data scientists. If you build just one product for everyone, you will have to build it for the average user. But in high dimension, in practice, no one has the characteristics of the average user. So you are building a product that perfectly fits no one! The entire idea behind data science and data products is to solve this statistical problem via personalization. And it is the difference between, say, FB newsfeed that is personalized on a user level using data science and a website that is the same for every user.

LinkedIn has tested a new UI with the goal to increase the number of likes per user. They test it by giving the new UI to a random subset of users

Test wins by 5% on the target metric. What do you expect to happen after the new UI is applied to all users? Will that metric actually go up by ~5%, more, or less? Assume there is no novelty effect here.

Answer:

This is again another way to ask about an A/B test where t-test independency assumptions are not met and check the implications.

Firstly, you should highlight that by randomly splitting users, test and control groups are not independent. And, therefore, test results cannot certainly be accurate.

It is important to emphasize once again why there is no independency between test and control. If a post is liked by a test user, this is seen by her connections in the control group as well. And control people might like that post as a consequence. Therefore, the new UI has an effect on both control and test users, with the potential of generating additional likes from both groups.

Here the test is winning by a very large margin. So it is safe to assume the new UI is better. If it is better, it is also safe to assume the new UI has generated additional likes in the control group, as should be clear from the example above. So control group numbers are likely inflated and it is likely that, if applied to all users, this

change will lead to a larger gain than 5% vs old UI.

Another reason to expect a larger gain is that social networks are affected by network effects (obviously). The more people are using a given product, the more value there is for other people in the network, and, therefore, the more likely other people are to use the product itself. This creates a virtuous circle.
Once this new successful feature will be given to everyone, its value will be higher than just a subset of users and, as a consequence, final metric gain will be higher than what a test would suggest.

Describe one example of a classification problem where the cost of a false positive is way higher than false negative as well as the other way round

Answer:

Whenever you tackle a data science project, one of the most important starting points is to have well clear whether the cost of a false positive is significantly higher, lower, or similar to the cost of a false negative. This will affect the model you choose, internal loss function, prediction cut-off point, and, possibly, even the training set itself.

A good example for large false positive cost is the recruiting process. In general, the cost of hiring a bad candidate is much higher than passing on a good one. A bad hiring will cost a lot of money in terms of not generating value, affecting the team morale, firing expenses, and the subsequent new recruiting process costs. Assuming you will have more than one good candidate in the pipeline within a reasonable time frame, passing on a good one is not that bad.

At this point, it would be interesting to draw a comparison between your current recruiting process and a machine learning classifier. You are interviewing with several people. Each interviewer can be seen as a classifier and the final classification will be given by a combination of each interviewer vote. This is exactly like ensemble methods. Furthermore, each interviewer will focus on a subset of your skills. Again, this is, for instance, similar to what happens in a Random Forest (RF). In a RF, each tree is built on bootstrap replicas of the original dataset. A consequence of this is that some events will get higher weight, so each tree can focus on learning more specific areas.

This is also a nice example since it will highlight how you consider the company where you are interviewing

particularly selective, which is always a nice thing to hear from the interviewer side.

A classical example of a large cost of false negatives is cancer detection. If you predict someone has cancer and it is not true, nothing major really happens. That person will have a bad few days until she finds out she is healthy. The opposite can have dramatic consequences, since that person will not get the appropriate treatment.

Finally, please note that minimizing the cost of false positives/negatives is a machine learning problem as much as a product problem. It is extremely important to re-design the product experience in a way that reduces those costs. A common and effective strategy is to design three paths: one for people with low probability of being case 1, one for people with high probability, and one for people in between.

For instance, in the recruiting case: the really good ones get an offer, the average and bad ones get rejected, and the good ones start with an internship. In risk, people predicted as "no fraud", go through the normal site experience. People predicted as "very high risk" get blocked. And people predicted as moderately high risk, go through an additional verification step live on the site.

How to calculate for how long I should run an A/B test?

Answer:

A super classic! You are almost guaranteed that during the recruiting process, you will be asked something like this. Sometimes they straightforwardly ask "How to estimate sample size in a t-test" and sometimes they do like in the question above. It is really the same.

It is extremely unlikely that they want the actual formula here (although it wouldn't hurt knowing it). Usually, they want to know which parameters you need to define in order to estimate the sample size. That is:

- Significance level, usually 0.05
- Power, usually 0.8
- Expected standard deviation of the change in the metric
- Minimum effect size you are interested in detecting via your test

Via those values, you will be able to get the sample size required for your t-test. Then, by knowing how many users you have per day, you can estimate how many days it will take to reach the required sample size with a 50/50 test-control split. If the final number is larger than 14 days, you are done. That's the number of days required.

If the final number is less than 14 days, you still want to run the test for 14 days in order to reliably capture weekly patterns. In this case, you might want to change the test-control split accordingly (like for instance 20% test and 80% control if 20% of users per day * 14 gives you the required sample size). That way you will still

get the required sample size in 14 days without "wasting test traffic".

Click here for mock phone interviews to practice similar questions, DS takehome challenges, and more

Suddenly, our dashboard shows that the number of picture uploads per day by Internet Explorer users went to zero. What could be the reason?

Answer:

If a metric drops it can be either a technical problem or a sudden change in user behavior. However, if a metric goes to zero for a given subset, it is definitely a technical problem. You can pretty much rule out the option that, suddenly, not one single user chooses to perform a given action.

So we can safely assume it is a bug. The question becomes which part of the code can have a bug. There are essentially 4 possibilities here:

- There is a bug on the site that doesn't allow to upload pictures via Internet Explorer browser
- Users can upload the picture normally, but it is broken the code that logs that action and stores it into the appropriate table in the database
- The action is stored in the right table, but there is a problem with the query that's supposed to access that table and return total number of uploads for Internet Explorer
- The bug is in the dashboard code that visualizes that metrics

After defining the possible root causes, you can go and inspect each of those 4 steps, until you find out what's broken. In general, the first option, i.e. user facing bug, is the easiest to catch. When users can't perform an action, they usually complain with customer service people right away. So you often end up knowing it even before having to test the code.

Indeed, a useful thing that some companies do to catch user-facing bugs as fast as possible and minimize their impact is:

- Build a tool that keeps scraping customer service tickets
- Use NLP to extract tickets related to bugs and understand their domain
- Automatically email them to the engineers in charge of that part of the product

Note that this question is a subquestion of the question: "We found a drop in pictures uploads. How to find out the reason?". A possible way to answer it is to say that, firstly, you segment users by various features, such as browser, device, country, etc. Then you assume that you discovered that one segment dropped to zero. So you say it is likely a bug and, finally, explain where the bug could be, as above.

LinkedIn has launched its first version of the People You May Know Feature. How would you isolate the impact of the algorithm behind it w/o considering the UI change effect?

Answer:

Whenever you launch a first version of a data product, i.e. a new product powered by machine learning, you are making a lot of changes on the site. Let's consider the People You May Know Feature. The first time it was launched, it implied adding to the user newsfeed a new box with clickable links. That new box with additional links by itself has high chances of moving the target metric, regardless of how good was the algorithm used to suggest people.
It is therefore hard to understand in which proportion the metric change was driven by the algorithm behind the new feature vs the UI change needed to accommodate the new feature.

In these cases, you need to test each component separately. After all, the whole point of A/B testing is to isolate the effect of just one change. A way to exactly isolate the two components is to run 3 versions of the site at the same time:

- Version 1 is the old version
- Version 2 is the site with the People You May Know Feature, where suggestions are based on the machine learning model that was developed
- Version 3 is the site with the People You May Know Feature, but suggestions are random

The difference between version 2 and 3 will tell you the gains coming only from the model.

This approach is risky though cause users in version 3 might lose faith in the feature, and once users decide a new feature is bad, it is really hard to make them change their mind.

A milder approach would be to replace random suggestions with something super basic, which machine learning should easily beat, but that still makes sense. For instance, you can use a history-based model (suggest users whose profiles were visited in the past by that user) or simply suggest users with the highest number of shared connections. These versions will still give a baseline to compare your model to, without giving users in one test group the impression that your new feature is very stupid.

How would you find out if someone put a fake school on LinkedIn? I.e. they actually didn't attend it

Answer:

There is a whole category of questions related to this: how to find out if users put wrong/fake information on a social network profile. And while this is mainly a fraud problem, it is not only a fraud problem. People can make mistakes (i.e. Rome, Georgia instead of Rome, Italy as hometown). A system that helps people to avoid mistakes can be really useful.

The safest approach to answer the question would be to ask to validate schools via the school email address. This will remove "frauds" (i.e. wrong schools), but it will also negatively affect legitimate users. Many people won't bother to use their school email address to validate it and, therefore, you will have a lot of missing values.

Whenever you have a social network and are trying to find some sort of information about a user, a really efficient approach is to look at their network. It is easy to put on a LinkedIn profile fake information, like a user went to MIT. It is way harder to build a network that can realistically resemble the network of someone who went to MIT.

An approach could be using anomaly detection. For instance, take all users who went to MIT. Extract information about the user themselves, as usual a combination of user info from their profile + how they interacted with LinkedIn. For instance, in this case, it would be very useful variables that show how many connection requests they sent, how they were distributed over time, acceptance rate, whether they visited other

people profiles before sending the connection request. Add variables about their connections, such as school, location, as well as their connection connections.

Now look for anomalies. You can build clusters and find either small clusters of different and suspect behavior (meaning everyone belonging to these clusters are putting fake schools) and/or you can find isolated points which are very far from legitimate clusters (meaning single users are behaving in a suspect way).

In practice, the most likely scenario is that if people put MIT without having gone to MIT one of the options below will be true and will be picked up as an anomaly:

- They have very few connections who went to MIT
- They have several MIT connections, but they had a very low acceptance rate, i.e. to get 100 connections from MIT people, they had to send way more connection requests
- Their connections are not connected to each other, meaning they randomly sent connection requests to other MIT users, instead of classmates or lab-mates who should be all connected with each other

As always, product re-design here can help as much as machine learning. For instance, you could let everyone put whichever school they went to. But if your anomaly model finds something weird, you could ask them to validate with the school email address. This way you would minimize the false negatives (missing values in this case) cause the majority of people would have no additional step. And would also minimize false positives (fake schools) cause "risky people" would have to validate the information. This would be exactly the same as the previous example of showing two-step-authentication only to risky people.

You are supposed to run an A/B test for 3 weeks based on sample size calculation. But after 1 week, p-value is already significant with test winning. So your product manager pressures you to stop the test and declare it a winner. What would you tell her? Explain in layman's terms

Answer:

Data scientists often have to communicate test results to people without a statistical background. Many interview processes check your ability to do that effectively by asking you to explain something related to a statistical test in "layman's terms" or "how would you explain this to a non-statistician".

To answer these questions avoid using expressions like p-value, type 1 error, NULL hypothesis, etc. The whole point is to explain those concepts in an understandable way to someone who just wants to know whether she should go ahead with the change or not.

Using analogies from everyday life is a good way to answer the "layman's term" questions. Since tests involve winning and losing, a good approach could be using sport, but really pick anything that you are familiar with, that has a win/lose outcome, and is not deterministic, i.e. if it happens multiple times you can expect different results. More in general, the characteristics above make a sport match a good choice whenever you have to explain frequentist concepts.

There are obviously infinite ways to explain in layman's terms why the product manager strategy is wrong. Let's pick soccer. The test was designed like a normal soccer game whose time duration was fixed in advance (i.e. 90 minutes for the game and 3 weeks for our actual test) and, at the end, you would check the result. Doing what the product manager is asking you to do would pretty much be like for a team to win the game if, at any point in time during the game, they happen to be ahead. This would obviously give that team a huge advantage, tremendously increasing the probability of winning, even if they are not actually better.

Note that this question is actually very practical. There are a lot of incentives/pressure within companies to declare a test a winner, for many obvious reasons. So being able to clearly articulate why you need to follow certain guidelines is really important.

What are the issues with splitting a small dataset (<1K events) in training/test set? What would you do then?

Answer:

You might have heard sentences like: in the last year more data was created than in the previous thousands of years or things like that. Despite being true that companies, and especially tech companies, are generating unbelievably huge amounts of data, that doesn't mean that there are not situations in which you will have too few data for your analysis. Matter of fact, it will happen quite often.

Larger amounts of data have led to trying to solve narrower and narrower questions. And as you try to narrow down the question, eventually trying to optimize for smaller and smaller segments of users, at some point you will end up with very few data. Beside that, if you work for a start-up, you won't have billions of users like Google or Facebook.

If your dataset is too small, the actual model will depend heavily on the random way in which you split the data into training and testing set. And that happens because the distribution of the training set from which the model learns will change significantly based on the random split. The distribution of training and test set can also be so different (i.e. each set has not enough events to converge to the true distribution) that whatever the model learns on the training set will be hardly useful to predict test set events; and, more importantly, to predict future events.

A common approach in this case is to cross-validate. I.e. split your data set into n (say 10) folds, then build 10 models. Each model uses a different combination of 9 folds to train and the remaining 10th to test. Final loss

can be estimated by taking the average of the loss of the 10 models. This way results do not depend on one single random test/training set split and, without overfitting, the entire original data set contributes to the learning step (unlike when you split once into training/test set. In that case, the test set is never used from a learning standpoint).

Another option when you have too few data is bootstrapping your original dataset. By bootstrapping you can simulate sampling more data from the original distribution.

Note that in both cases the techniques work well if the dataset is small, so you'd rather not split it, but it still has reliable information about the true distribution. If your dataset has not enough information to answer a given question, there is not much you can do beside figuring out how to collect more data or changing the question. You can bootstrap a dataset of 1 row 1 million times and get a 1MM row dataset, but won't be very useful!

Using LinkedIn data, how would you predict when someone is going to change job? Assume you can use all LinkedIn user activity data

Answer:

This kind of question is pretty straightforward and should be very much expected if you are interviewing at LinkedIn.

As usual, let's start by defining the label. Especially, at which point in time do you want to make the prediction? Let's say you want this prediction to be done on a monthly basis. At the end of each month, you take a snapshot of user data up to that month, and predict for all LinkedIn users the probability that they will change job in the following month.

Regarding which variables to include, you will have a combination of **user profile data, data about when you took the snapshot, user behavior on the site, and some external data about job demand**.

1) For user profile, possible variables are: school, major, previous companies, job titles, job level, how long they have been at their previous/current companies, when they graduated, skills, relevant keywords from job description.

2) You also need a variable about the month you took a snapshot with all user information. This is important because, for instance, people tend to change job more at the beginning of the year, for several reasons (such as that's when bonuses are paid and companies have more hiring budget available).

3) User behavior. Assuming you can collect all information from LinkedIn usage, this would include last time

they have updated their job descriptions, skills, or asked for recommendations. Number of connection requests, especially to HR or hiring mangers and, if available, even variables related to searching for jobs in the specific LinkedIn section.

4) Finally, integrate this with variables related to demand about jobs in a similar field/level of the LinkedIn user. A sudden spike in demand will mean stronger negotiation power and, therefore, creates an incentive in changing job.

This problem is a standard binary classification problem. 1 means changing job and 0 means the opposite. Collect data about users, say ranging from 1 to 25 months back, and add a label whether the user changed job in the following month or not. When you collect the data, make sure that, if you are trying to predict if someone will change job in, say, Feb. 2017, you only have user data up to the end of Jan. 2017.

Pick any binary model. A Random Forest (RF) is likely the safest choice because I am expecting here high dimension, highly correlated variables, and outliers. RF works well in those cases. After building and testing the model, apply it to future data. The forest output, i.e. the percentage of trees that classify an event as 1, can be seen as probability of changing job.

Btw note that if you did have all those variables available, it is really likely that the behavioral ones will be way more informative than the others. And should be pretty obvious why. Similarly to the discussion about advertising, variables about user behavior (like what and when they search for something, where did they click, etc.) are more powerful than variables about user demographic (like school, current job, etc). That's because user behavior variables tell us the moment when they are planning to do something. The time part is crucial.

Between the following two metrics, which one would you choose to measure response time of an inquiry at Airbnb: percentage of responses within 16 hrs or average response time considering only responses within 16 hrs?

Answer:

The goal of a metric is to have something measurable that's positively related to a behavior you want to incentivize and negatively related to something you want to disincentivize. That way the metric is a key indicator of whether things are moving in the right direction or not.
For instance, vanity metrics are bad cause they tend to go up no matter what and, therefore, they are not particularly informative.

As a general rule, metrics based on the percentage of users who perform a certain action or, similarly, based on percentiles tend to be more robust. Firstly, they are not affected by outliers. Secondly, even in cases where not all the users are performing a given action, they can still be evaluated including the whole population. To clarify this, let's look at the question: not all hosts respond. If we take the average then, we can only include those hosts who did respond within a certain time frame. By taking the percentage who responded within 16 hrs, we can include everyone in the formula, regardless of whether they responded or not.

In 99% of the cases, when they ask you to choose between two metrics, there is a clear catch: either one metric takes too long to age so you can't realistically use it, or it is not always positively correlated to a behavior

the company wants to incentivize.

In this case, there is a problem with taking the average. Let's assume for simplicity that the average response time, considering only responses within 16 hrs, is 8 hrs. In this scenario, the metric will go up if hosts who used to respond in say 12 hours stop responding. Optimizing for that metric will mean finding ways to make not respond hosts who used to respond in more than the average. And any Airbnb user will tell you that it's better to get a response in 12 hrs than no response.

On the other hand, focusing on percentage of users who respond within a certain time frame will imply focusing on improving response time for those who currently are outside of that threshold. This metric is clearly better than the other one, but it also has some cons. Specifically, the metric doesn't change whether hosts respond in 1 minute or 15 hrs. They are still counted as success cases when we take the percentage. It doesn't push us to optimize for excellence (i.e. super fast responses), but focuses on giving everyone a good enough experience.

In cases like this, it is common to have a couple (or more) of metrics with different thresholds and try to optimize for both at the same time, or at least making sure that, if one goes up, the other one doesn't go down. One represents excellence with an extremely aggressive thresholds and one with a larger threshold is meant to capture good enough experience (say responses within 1 hr and 16 hrs).

Finally, please note that there are businesses that are largely based on outliers. That is, the high high majority of their revenue comes from few power users. Online gaming is an example. In those cases, you do want a metric which is affected by outliers. We will see an example below.

At FB, we found out that users with filled out profile infos (age, hometown, etc.) are more engaged than those without.

Therefore, we figure out a way to fill out those infos automatically for all users hoping it would improve engagement. However, engagement barely changes. Why?

Answer:

This is another classic and you will find it rephrased in endless ways. Another common version is using the profile picture -> engagement relationship. The characteristics of this category of questions are:

- Give one or more variables which are positively related to the output (here profile information -> engagement). These variables are always about a choice specifically made by the user. That is, in this case the user can choose to give or not some information about herself
- Create a situation where you did manage to positively affect those variables (profile info here), but this had no effect on the output (engagement)

So, these questions can really be reduced to: we know A is related to B, but after we managed to move A, B doesn't change, so A wasn't causing B.

Let's start from an example to make this clearer. For sure, FB users who fill out their workplace are more engaged than those who don't. And let's imagine FB runs an algorithm to scrape the web and find where its users work (easily doable) and automatically fills that information out. Would this significantly increase engagement for those users? In most cases, the answer is no.

The reason being that filling out profile info is a proxy itself for engagement. Users self-select themselves based on their engagement level when they choose whether to fill out a profile or not. Increasing engagement via other means will likely increase the probability of filling out the profile. But the opposite is hardly true.

That doesn't mean that discovering that people with no profile info are less engaged isn't useful. It is actually super useful, but the approach needs to be different. Instead of running a test to increase that number, a better approach is:

- Make the proxy for engagement your output. I.e. label people with no profile info as 1 and others as 0. You are essentially defining engagement here
- Select variables that can relate to that output, as usual demographic/user and behavioral/browsing characteristics. Focus on actionable variables
- Build a model to predict your label and identify, among the most important variables, the ones that are most likely to have a causal relationship with the outcome
- Figure out how to positively affect those variables

And, btw, this is exactly the same approach as the answer how to improve engagement on FB. Only difference here is that we used as a proxy for engagement profile info.

When answering any kinds of questions, always think about whether (self-)selection biases are there or not. They are extremely common in tech. And btw it is also common for companies to send takehome challenges where one segment (i.e. no picture) performs much worse and they ask to explain why, the reason simply being self-selection bias.

You ran an A/B test last year and it lost. When would it make sense to re-run the same test today?

Assume there were no bugs in the test, everything was statistically as well as technically sound, and the site experience is the same.

Answer:

If the test and current site are the same, there were no issues with the test last year, and technology is the same, the main case in which I can realistically expect different results is if the underlying distribution of users has changed.

A change in user distribution is something that happens quite often (or at least should happen!). Early users are more tech-savvy and have in general different characteristics than the new users a company will acquire later on in its life. There is no reason to assume that a test that failed last year should also fail this year, if user base has evolved.

It is therefore pretty common to look back at past tests and see if it makes sense to re-run one of them. A frequent way to see if it makes sense to re-run a test is to look at past test results, change the underlying distribution of users giving weights so that it resembles the current distribution, and see if results are promising.

Note that any kind of analysis you do on past test results, but asking a different question, can only be used to

inform the decision of running new tests. You can't ask a different question to an old test and, if the test wins under the new question, declare it a winner. You would end up with many false positives (if you torture the data long enough, it will confess to anything).

You might also get slightly different questions along the lines of: "What's the major drawback of A/B testing?". They are all hinting at the same thing as the question above. The fact that A/B test results are not telling you in absolute terms which version is better. They are telling you which version is better given your current user base, which is the data you use to test.
Matter of fact, it can be dangerous for a small company to over optimize using A/B testing on the entire population and to only look at the overall results. They might end up with a great site for tech-savvy and early adopter people, but never grow beside them, essentially finding local optima.

How would you identify if an advertiser is using clickbait techniques without having a dataset with labeled events?

I.e. you don't have a dataset that tells you for the past ads which ones are clickbaits and which ones are not

Answer:

Clickbait ads are those advertisings with a very catchy headline that induce people to click on them. But the destination page after clicking has little to do with the headline.

Defining from a metric perspective clickbait ads, we can expect a significantly larger than average click-through-rate (CTR). This comes from the catchy headline. However, we can also expect a sharp decrease in CTR over time for those users who clicked on them. These users have lost faith in the ads and, therefore, will click much less on them in future.

Please, note that this answer is about ads CTR, but, as should be clear from the definitions above, can really be applied to any questions of the family: "How to know if a short term change is positive, but will be negative in the long term". For instance, sending emails or notifications to users is an example of this. If you suddenly increase the number of notifications, you will likely get a short term gain followed by a long term drop.

To identify the bad ads, we can directly use the metric definitions above. For each ad, take that given ad CTR +

medium term (say 2 weeks) change in CTR for users who clicked on that ad. Cluster ads on these two dimensions and identify the cluster(s) whose centroid has a high value for CTR and low for CTR change. Ads assigned to that cluster are the clickbait ones.

Once you have the labels from the previous step, if you want to block future clickbait ads before the user sees them, it becomes a standard supervised machine learning project. For each labeled ad, pick all sorts of variables related to the ad, the text, as well as the advertiser, and build the model.

Finally, note that this answer can be adapted for different kinds of questions. If the question were: find the worst performing ads (not just clickbait ads, but all ads bad for the business) or the best ones, you could go back to the clustering step and simply take a different cluster, based on whatever the question is asking for.

What are the most important parameters in a Random Forest?

Answer:

Not a product question, but it is so common that it is important to include it. In large tech companies, job interviews are largely focused on product and metrics. However, you can expect a few theoretical questions. Theoretical questions will most likely touch either Random Forest (RF), logistic regression, or t-tests. It is extremely unlikely to be asked anything beside those things, if you interview for a product data science role for a large tech company.

The most important parameters in a RF are:

- The number of trees. In general, as you start increasing the number of trees, accuracy will improve. Up to a point where accuracy stops improving, as you add more trees. That number is the optimal number of trees. If you plot on the y-axis loss (for instance 1 - accuracy) and x-axis is the number of trees, the x-axis value when that curves flattens is the optimal tree number. Often, that number will be somewhere between 50 and 200. The larger the number of trees, the slower the training and testing time (although the trees can be built in parallel, so you can significantly reduce processing time with a wise implementation).
- Minimum number of events per tree. A RF benefits from large trees. Generally speaking, the larger the tree, the better it is from a classification standpoint, although it means a slower model. Start from trees as large as possible (i.e. 1 minimum event per leaf) and increase that value as long as you are not losing accuracy. Stop when your model start performing significantly worse. If you are not concerned about processing time, just set it as 1.
- Number of features randomly sampled at each split. Beside building each tree on bootstrap replicas of the original dataset, an additional way to build trees that are different among each other (so that they can

capture different information) is to not split on the best variable, but to split on the best variable only considering a different random subset of variables at each split. The optimal number for this parameter varies greatly depending on the data. Most common values tend to be between square root of n and n/2, where n is the number of features. However, as often in machine learning, it is one of those parameters where you just try different values until you find the best one for your problem.

In on-line gaming companies, do you expect the average revenue per user to be larger or smaller than the median revenue per user?

Assume the following business model: game can be downloaded and played for free and then there are in-app purchases.

Answer:

A common business model is to give away a product for free and then charge for upgrades or specific additional features inside the product. This is sometimes called freemium and is particularly common in the gaming industry.

The median is smaller than the average if the distribution is skewed to the right. In the likely scenario that more than 50% of my users only use the free content, automatically the median would be smaller than the average, as long as at least one user is buying anything.

The majority of business metrics are skewed to the right, especially in the freemium model. The high majority of users will just use the content for free, and a few users will spend a lot of money. These are outliers on the right side of the distribution. And outliers move up the average significantly. However, median is hardly affected by outliers. So the median will be smaller than the average. Note that this happens when I have outliers only on one side of the distribution. Here the distribution is cut at zero, so I can't have negative outliers to balance the positive ones and bring down the mean.

To put this into perspective, King.com reported that half of its revenue, and we are talking about billions of dollars, was generated by only 0.15% of its users. Since the high majority of money comes from few people, these are cases where you do want a metric that is sensitive to outliers. Similarly, there might be cases where you actually want a model that is sensitive to outliers, so trees wouldn't be a good choice there.

How would you increase revenue from advertising clicks if you were working for an ads company (i.e. Google, FB, etc.)?

Companies get paid when users click on ads.

Answer:

There are obviously infinite ways to answer this question. One of the most straightforward is:

- Break down revenue from ad clicks into its components. Revenue depends on:
    - Ad click through rate (CTR), i.e. probability of users clicking on an ad when they see it
    - Total number of times ads are shown to a user
    - Cost per click, i.e. how much money each click costs

- After this, find something that moves one of those components in the right direction without negatively affecting the other ones. This is much safer than trying to guess scenarios where one component goes up and another goes down, but the positive change outweighs the negative one

Some examples could be:

Increase CTR by better targeting. Build products that allow to collect better and more specific data about user intent (obviously users should benefit from these products too, or this will negatively affect number of users in the long run and, therefore, revenue will drop). Or, assuming data doesn't change, improve CTR predictive

models with more advanced techniques.

Increase number of page views. This can be achieved in several ways. One is faster browsing. Everything else being the same, it allows for more page views (and this is the number one reason Google benefitted so much from Chrome).

Another example is, obviously, increasing the number of users or increasing page views per user. For instance, web pages that automatically update their content as you scroll down, like newsfeeds, can be seen as a way to increase page views. They are essentially removing friction from browsing through several pages.

Finally, you should also focus on maximizing probability of a user converting on the advertiser site. Eventually, advertisers will pay for an ad click its true value (this is especially true given that ad bidding is typically based on second-price auctions. And in those cases the bidder has an incentive in bidding the true value).

The true value of an ad is the average value of a user coming to their site from ads. If you can increase that value, advertisers will pay more. This can include focusing on probability of conversion on the advertiser site over just CTR in your ad models, or working with the advertisers to improve the user flow after people click on an ad.

Give an example of a site change that we can't test on a subset of users via a controlled experiment. How would you estimate the impact of that change?

Answer:

The high high majority of changes that happen on a site are firstly tested on a subset of users. This is typically done either by splitting users randomly or by using some markets as test group. However, not all changes can be tested in advance. For instance, when a company launches a new logo or redesign the entire site experience, this cannot be tested in advance. You have to launch a new logo for everyone on the same day.

Understanding the impact of these kinds of changes is, honestly, more an art than a science. Breaking down the steps using the logo example:

The first step is usually doing qualitative research. The new logo is shown in real life or via a survey to a small group of users and the feedback is gathered. You can look at this as a mini-A/B test, where the sample size is too small to give statistically significant results, but it can still be directionally informative.

After the new logo has been chosen and launched, a data scientist is asked to compare actual data after the launch vs predictions, if nothing had changed. That is, let's say your company number one metric is conversion. You build a time series model for conversion using, in your training set, all the data until the day before the change. This model then predicts what would have happened without the change. The delta between predictions and actual numbers observed is your estimate of the impact of the new logo.

There are a couple of issues with this approach:

- You are comparing predictions vs actual numbers. Predictions are, by definition, noisy and not 100% accurate. So one of the two groups you are using in your comparison is extremely noisy, affecting the final results.
- If small changes are affected by novelty effect, large changes are affected by change aversion. This is exactly the opposite of novelty effect. Change aversion means that, if you change something very important, users won't like it at first, until they get used to it.

For the first point, i.e. noise coming from the predictions, there is not much you can do. However, it is relatively safe to assume that the effect of the change should clearly be seen despite the noise, if successful. These are not small changes, otherwise they would be tested the normal way. Therefore, the results of the comparison between predictions and actual numbers might not be super accurate, but should be directionally true (i.e. metrics went up or not?).

Regarding change aversion, if you have past data about large changes, you can estimate it and then account for it in your comparison between predictions vs actual numbers. This is what typically happens in large companies, where change aversion has been modeled thanks to past data.
If you have no way to model change aversion (either via internal or external data), you will need to wait a longer time frame before being able to estimate the actual effect of the change.