Predictive Analytics
Drilling sense out of data

Interviewer **Framework**

When interviewing for a data scientist position, you need quickly discover a lot about the candidate. An ideal data scientist's skillset spans

- Math
- Statistics
- Programming & databases.
- Business expertise.

You need to probe each of these areas. You also want to find out if they are smart and get things done.

You typically only have an hour to get a feel for how a candidate fares in each of those areas followed a week or two later by a case study if you think the candidate has potential.

Introduction

Give them an overview of your company and your honest pitch on why your analytics team is great. Ask them about themselves to get a feel for their background, and ask them what they are looking for in a job. If you do this right, the candidate is excited for the rest of the interview.

Description of a recent problem

You to hear about a project they've worked on recently. Ask them about how the project started, how they determined it was worth time and effort, their process, and their results. Also ask them about what they learned from the project. You gain a lot from answers to this question: if they can tell a narrative, how the problem related to the bigger picture, and how they tackled the hard work of doing something.

Technical deep dive

Check if they have skills in statistics, databases, programming, and interpreting data. (More of this later)

Ask them for questions

There are so many possible questions they could ask that if they don't have any questions it's a warning sign that they don't think before they act. Not having questions could also be a sign that they don't want the job.

Technical questions

An interview has one purpose: to see if this person will be successful in the role you're offering. From a technical standpoint, that means checking they have the prerequisite knowledge for the job. But people are dynamic creatures who learn and grow, and if a person is missing knowledge they can go read Stack Overflow and pick it up. So a technical interview shouldn't be a test of exactly how much they know on a topic from memory.
The interview questions are guided by three principles:

No trick questions or tests of cleverness.
No question should require a candidate to get to an "a-ha" during the interview. You should only test them on things they should feel comfortable answering with their existing knowledge. A brain teaser question like "suppose you have a stack of pancakes in a random size order, how would you take a spatula and order them in the minimum number of moves?" doesn't relate to what doing data science is in practice. Further, many strong candidates may not be able to answer this in seconds, because they need time to think and process. These sorts of questions generally are there to make the interviewer feel clever for knowing the answer!

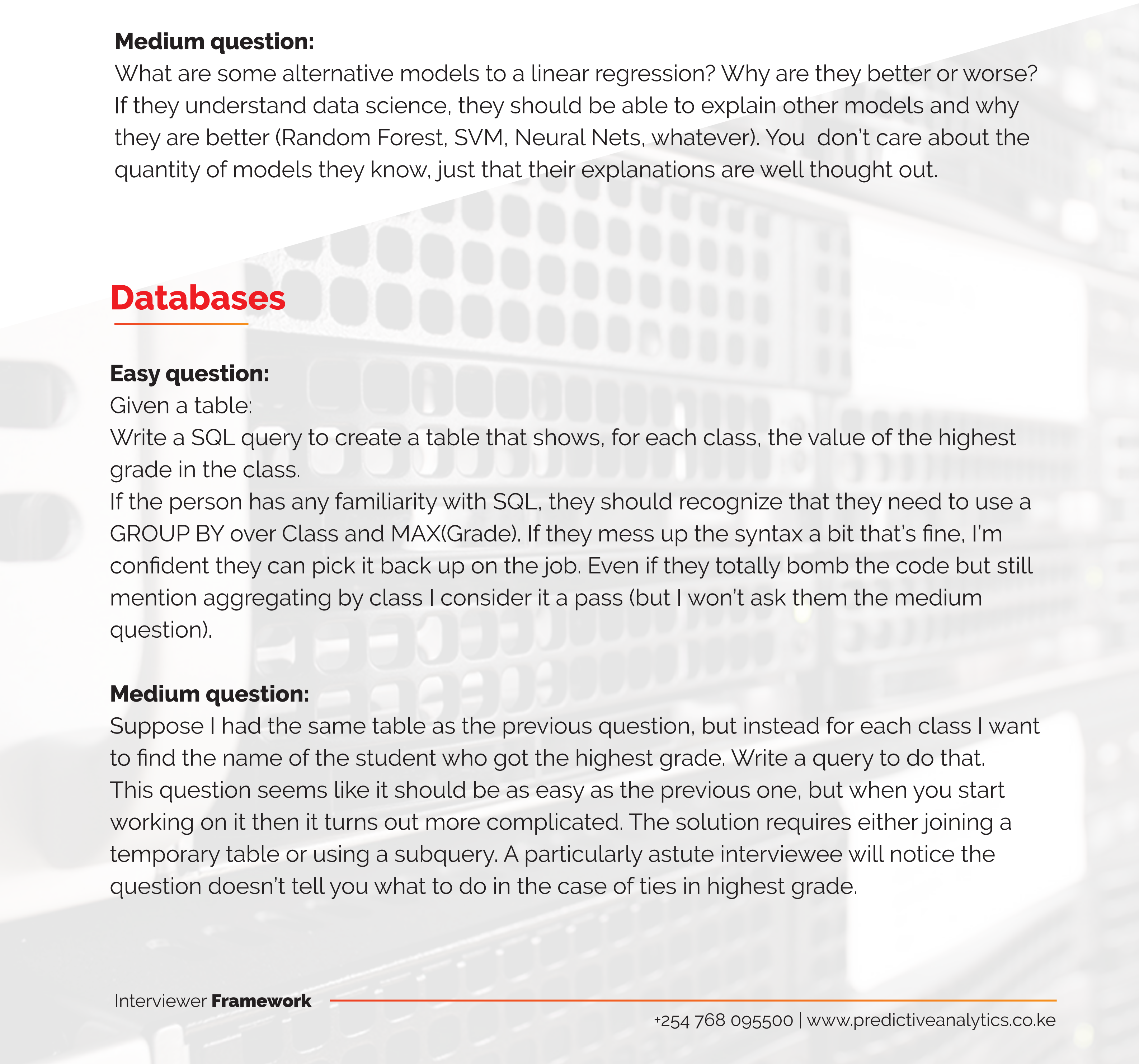
Only test on the basics.
Given a particular area, only ask questions at an introductory level. For instance, if asking a question about machine learn models, only ask about linear and logistic regressions and avoid asking about more advanced topics like Random Forests or boosting. The reason for this is that if they understand the basics they should be able to pick up the advanced topics on the job. Further, as you get into more advanced topics there is a higher likelihood that the candidate just never happened to deal with that topic. If they do know a lot of advanced materials, that will likely be noticeable in how they answer the basic questions.

Keep questions a discussion.
Try to avoid questions where there is a single right answer, because the only information you get is if the candidate knows that exact answer or not. For instance, instead of asking "what is the difference between a left join and an inner join" You would ask "what are joins in SQL" and if they give a decent answer you would ask "what are some different types of joins?" The candidate may come up with a more interesting answer and You can probe into times they've had tricky joins to do.
For each area, you first ask them their familiarity with the topic. If they say they don't have much, You skip it. You want to avoid having the candidate feel overwhelmed or frustrated by that topic, as that could jeopardize the rest of the interview.

Statistics

Easy question:
How would you explain a linear regression to a business executive?
This question tests if they have a good mental model of what a linear regression is, and if they can explain it in non-technical terms. The common way people mess it up by being too technical "suppose we have normally distributed errors in our dependent variables." You are looking for an answer that sounds something like "it's a way of predicting a value as being proportional to some other values" along with a simple example.

Medium question:
What are some alternative models to a linear regression? Why are they better or worse?
If they understand data science, they should be able to explain other models and why they are better (Random Forest, SVM, Neural Nets, whatever). You don't care about the quantity of models they know, just that their explanations are well thought out.



Databases

Easy question:
Given a table:
Write a SQL query to create a table that shows, for each class, the value of the highest grade in the class.
If the person has any familiarity with SQL, they should recognize that they need to use a GROUP BY over Class and MAX(Grade). If they mess up the syntax a bit that's fine, I'm confident they can pick it back up on the job. Even if they totally bomb the code but still mention aggregating by class I consider it a pass (but I won't ask them the medium question).

Medium question:
Suppose I had the same table as the previous question, but instead for each class I want to find the name of the student who got the highest grade. Write a query to do that.
This question seems like it should be as easy as the previous one, but when you start working on it then it turns out more complicated. The solution requires either joining a temporary table or using a subquery. A particularly astute interviewee will notice the question doesn't tell you what to do in the case of ties in highest grade.

Programming

Easy/medium question:
In pseudo-code or whatever language you would like: write a program that prints the numbers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".
Yes the classic FizzBuzz, known as the question that any capable software developer should be able to answer. You can find lots of writing on it, including a great enterprise version. Since data scientists are weaker programmers than software developers, this question is excellent for interviewing.

When a person is working on their problem, I listen to how they reason. If they make statements like "hmm, well I should probably look at each number so I'll use a for loop" or "I'll handle this by using an if-else statement — oh! I guess I need to check for Fizz-Buzz first since both the other conditions apply" that's a great sign. Provided they stumble to any working answer I consider this an easy pass.

To pass at the medium level, You ask them to improve their code by making so that the number/word pairs are part of the input, and I could pass an arbitrary amount of them (for instance you could add that 17 prints Jazz). A good-enough-for-data-science programmer should be able to do this.

You also ask them if they've heard of this problem before, since if they've read up on software development they likely would have heard of it already.

Interpreting data

Easy/Medium question:
A company selling a competitor to Microsoft Office is testing their marketing by sending out two different sets of emails. One set contains business related content, and one contains consumer related content. We are interested in how each campaign performed; did one do better at getting people to click-through? Below is a selection of graphs on the two email campaigns. The bottom two graphs have the same data as the top two, only bucketed by the amount the customer has spent with the company the year before the emails were sent. Which campaign did better?