1．**(Given a Dataset) Analyze this dataset and give me a model that can predict this response variable.**

Response variable: Variables of interest in an experiment (those that are measured or observed) are called response or dependent variables.

**2. What could be some issues if the distribution of the test data is significantly diﬀerent than the distribution of the training data?**

This is a problem of dataset shift. Some reasons are as follows:

1. Covariate shift: training and test input follow different distributions, but functional relation remains unchanged. *[Covariate Shift, when the distribution of the inputs used as predictors (covariates) changes between training and production stages; Dataset Shift, when the joint distribution of inputs and the output (the target being predicted) also changes.]*

2. Sample selection bias: the training examples have been obtained through a biased method, such as non-uniform selection. How to fix? **Cross Validation**: divide training samples into K groups; Train a learning machine with k-1 groups; validate the trained machine using the rest; Repeat this for all combinations and output the mean validation error. CV is almost unbiased without covariate shift. But, it is heavily biased under covariate shift.

3. Non-stationary environments: Training environment is different from the test one, whether it's due to a temporal or a spatial change. One typical scenario is adversarial classification problems, such as spam filtering and network intrusion detection.

**3. What are some ways I can make my model more robust to outliers?**

Here are some changes you can make to your model:

1. Use a model that is resistant to outliers. **Tree-based** models are generally not as affected by outliers, while regression-based models are. If you are performing a statistical test, try a **non-parametric** test instead of a parametric one.
2. Use a more robust error metric. Switching from mean squared error to **mean absolute difference** (or something like Huber Loss) reduces the influence of outliers.

Here are some changes you can make to your data:

1. Winsorize your data. Artificially cap your data at some threshold.
2. Transform your data. If your data has a very pronounced right tail, try a log transformation.
3. Remove the outliers. This works if there are very few of them and you’re fairly certain they’re anomalies and not worth predicting.

**4． What are some diﬀerences you would expect in a model that minimizes squared error, versus a model that minimizes absolute error? In which cases would each error metric be appropriate?**

Basically MAE is more **robust to outlier** than is MSE. MAE assigns equal weight to the data whereas MSE emphasizes the extremes – the square of a very small number (smaller than 1) is even smaller, and the square of a big number is even bigger. MSE has nice mathematical properties which makes it easier to compute the gradient. However, MAE requires **more complicated** tools such as linear programming to compute the gradient.

**5． What error metric would you use to evaluate how good a binary classifier is? What if the classes are imbalanced? What if there are more than 2 groups?**

Accuracy:

Definition – proportion of instances you predict correctly

Strengths – Very intuitive and easy to explain

Weakness – works poorly when the signal in the data is weak compared to the signal form the class imbalance. Also, you cannot express your uncertainty about a certain prediction.

Area under the curve (AUC)

Definition (intuitive) – Given a random positive instance and a random negative instance, the probability that you can distinguish between them.

Definition (direct) – The area under the ROC curve (a receiver operating characteristic curve is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. y: TPR, x: FPR)

Strengths – works well when you want to be able to test your ability to distinguish the two classes.

Weaknesses – You may not be able to interpret your predictions as probabilities if you use AUC, since

The AUC has several equivalent interpretations:

The expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

The expected proportion of positives ranked before a uniformly drawn random negative.

The expected proportion of negatives ranked after a uniformly drawn random positive.

The expected false positive rate if the ranking is split just after a uniformly drawn random positive.

LogLoss/Deviance

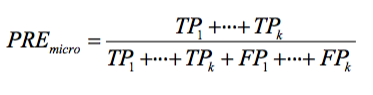
Strengths – Your estimates can be interpreted as probabilities.

Weaknesses – if you have a lot of predictions that are near the boundaries, your error metric may be very sensitive to false positive or false negatives.

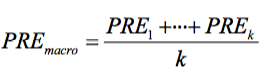
F-score, Mean Average Precision, Cohen’s Kappa

There are more esoteric and not so often used for general binary classification tasks. You may see them in specific subfields (e.g. F-score in NLP and Precision metrics in information retrieval)

To generalize this to multi-class, assuming you have a One-vs-All (OvA) classifier, you can either go with the "micro" average or the "macro" average. In "micro averaging," you'd calculate the performance, e.g., precision, from the individual true positives, true negatives, false positives, and false negatives of the the k-class model:



And in macro-averaging, you average the performances of each individual class:



**6. What are various ways to predict a binary response variable? Can you compare two of them and tell me when one would be more appropriate? What’s the diﬀerence between these? (SVM, Logistic Regression, Naive Bayes, Decision Tree, etc.)**

There are a number of dimensions you can look at to give you a sense of what will be a reasonable algorithm to start with, namely:

Number of training examples

Dimensionality of the feature space

Do I expect the problem to be linearly separable?

Are features independent?

Are features expected to linearly dependent with the target variable?

Is overfitting expected to be a problem?

What are the system's requirement in terms of speed/performance/memory usage...?

Logistic regression

* robust to noise and you can avoid overfitting and even do feature selection by using L2 or L1 regularization
* well-behaved classification algorithm that can be trained as long as you expect your features to be roughly linear and the problem to be linearly separable. You can do some feature engineering to turn most non-linear features into linear pretty easily.
* the output can be interpreted as a probability. This is something that comes as a nice side effect since you can use it, for example, for ranking instead of classification.
* set your baseline with Logistic Regression and next step can be SVM or Tree Ensembles. (preferable when learn nothing)

SVM

* SVM use a different loss function (Hinge) from LR. They are also interpreted differently (maximum-margin). However, in practice, an SVM with a linear kernel is not very different from a Logistic Regression. The main reason you would want to use an SVM instead of a Logistic Regression is because your problem might not be linearly separable. In that case, you will have to use an SVM with a non-linear kernel (e.g. RBF). The truth is that a Logistic Regression can also be used with a different kernel, but at that point you might be better off going for SVMs for practical reasons.
* Another related reason to use SVMs is if you are in a highly dimensional space. For example, SVMs have been reported to work better for text classification.
* painfully inefficient to train. So, I would not recommend them for any problem where you have many training examples. I would actually go even further and say that I would not recommend SVMs for most "industry scale" applications.
* Picking/finding the right kernel can be a challenge
* Results/output are incomprehensible
* No standardized way for dealing with multi-class problems; fundamentally a binary classifier

Tree Ensembles

* One main advantage is that they do not expect linear features or even features that interact linearly. Something I did not mention in LR is that it can hardly handle categorical (binary) features. Tree Ensembles, because they are nothing more than a bunch of Decision Trees combined, can handle this very well.
* The other main advantage is that, because of how they are constructed (using bagging or boosting) these algorithms handle very well high dimensional spaces as well as large number of training examples.
* Random Forests (RF) and Gradient Boosted Decision Trees (GBDT),

Instance based learning techniques (e.g. nearest neighbor)

* Take up a lot of memory to run (storing all the instances)
* Work well for a small number of dimensions, but not a high number of dimensions

Decision trees and divide-and-conquer algorithms

* May overfit data
* May get stuck in local minima so need ensembles to help reduce the variance

Neural networks

* Picking the correct topology is difficult
* Training takes a long time/requires a lot of data
* Output/issues are incomprehensible

Naive Bayes

* Very simple representation doesn't allow for rich hypotheses
* Assumption of independence of attributes is too constraining (although, see Domingos' paper on why it's not so bad)

Bayes networks/Graphical models

* Determining the topology of dependence is difficult

**7. What is regularization and where might it be helpful? What is an example of using regularization in a model?**

Regularization is a technique used in an attempt to solve the **overfitting** problem in statistical models.

You penalize your loss function by adding a multiple of an L1 (LASSO) or an L2 (Ridge) norm of your weights vector w (it is the vector of the learned parameters in your linear regression). You get the following equation:

L(X,Y)+λN(w) (N is either L1, L2 or any other form) How can I tune in the regularization term λ? One possible answer is to use cross-validation.

L1 is good for sparsity, when there are many inputs and you believe that only a few of them are meaningful. L2 is good at dealing with correlated inputs. If two inputs are perfectly correlated L2 put half of the weight (beta) on each input, while L1 would pick one randomly (so less stable). One can use a combination of L1, L2 to get a balance of both, also known as Elastic Net.

**8. Why might it be preferable to include fewer predictors over many?**

This reduces your risk of overfitting. If you have many predictors it becomes easier to fit your training data well but have a model which does not generalize well to a larger data set.

Occam's Razor principle: use the least complicated algorithm that can address your needs and only go for something more complicated if strictly necessary.

**9. Given training data on tweets and their retweets, how would you predict the number of retweets of a given tweet after 7 days after only observing 2 days worth of data?**

run GLM using the features. Features: both content and contextual features. Content features include the number of URLs and hastags and contextual features include the number of followers and followees and the age of the account. The number and frequency tweets and the number of favorited tweets seem not to matter much.

**10. How could you collect and analyze data to use social media to predict the weather?**

Just a simple thought:

Collect historical data:

Collect the historical feed of interest. This could be based on location or social network.

Define and label the weather in the past x days for each feed.

For each feed, find the words or phrase using with text matrix. There may be some featured terms, like ‘umbrella.… This could be a very sparse table, but feature reduction can be performed based on the weather information (only choose the highly frequent ones for a particular weather).

Other possible attributes could be date (month, quarter), location, etc.

With the new feed:

Extract the text and other attributes from the feed, and then data manipulation

Perform KNN with the historical data for a general weather prediction

<https://www.quora.com/How-could-you-collect-and-analyze-data-to-use-social-media-to-predict-the-weather>

**11. How would you construct a feed to show relevant content for a site that involves user interactions with items?**

Facebook: there are three main parts to EdgeRank called affinity, weight and time decay. Affinity is your relationship with users. It means that the amount a user has interacted with your Page in the past affects how likely they are to see your posts. Weight is how much priority EdgeRank gives to your post, based on the post type. Time decay simply means how old is your post? The longer your post has been on Facebook, the less likely it will be to show up in a user’s News Feed.

<https://www.quora.com/How-would-you-construct-a-feed-to-show-relevant-content-for-a-site-that-involves-user-interactions-with-items>

**12. How would you design the people you may know feature on LinkedIn or Facebook?**

Linkedin People You May Know product (Aka Connection suggestion algorithm) uses many features to calculate a connection probability among two people.

Some examples are: companies position overlap (How many months/years both individuals work together at same company, similar/related field) , school education overlap, strong unconnected people in linkedin connection graph. (both individual have many common friends) etc.

13. How would you predict who someone may want to send a Snapchat or Gmail to?

14. How would you suggest to a franchise where to open a new store?

**15. In a search engine, given partial data on what the user has typed, how would you predict the user’s eventual search query?**

The secret source is a model trained on a set of features and to weigh them differently even after each iteration, generating a list of ranked results. Some features might be:

* Popularity (collective intelligence). For example, if many people type in “new york” then go on type “new york times”, that can help make it a suggestion next time you type in “new york”.
* Region. If your set location in Google to be “Toronto” most likely “Toronto” will appear after you type in “pizza”.
* Search history. If you have searched “classical music” before, next time you type in “classical”, “music” might follow.

16. Given a database of all previous alumni donations to your university, how would you predict which recent alumni are most likely to donate?

17. You’re Uber and you want to design a heatmap to recommend to drivers where to wait for a passenger. How would you approach this?

18. How would you build a model to predict a March Madness bracket?

<https://www.quora.com/How-would-you-use-predictive-modeling-to-build-a-March-Madness-bracket>

http://blog.smellthedata.com/2013/03/now-that-march-madness-is-officially.html

19. You want to run a regression to predict the probability of a ﬂight delay, but there are ﬂights with delays of up to 12 hours that are really messing up your model. How can you address this?

There are actually three broad categories of feature selection algorithms: Filter, wrapper, and embedded methods.

Wrapper methods measure the performance of features based on the classifier, the "usefulness" of features if you will. In contrast, the filter methods pick up the intrinsic properties of the features (i.e., the "relevance" of the features) measured via statistical tests instead of cross-validation performance. So, wrapper methods are essentially solving the "real" problem (optimizing the classifier performance) but they are also computationally more expensive compared to filter methods due to the repeated learning steps and cross-validation.

The third class, embedded methods, are quite similar to wrapper methods since they are also used to optimize the objective function or performance of a learning algorithm or model. The difference to wrapper methods is that a intrinsic model building metric is used during learning.

Although you only asked for filter methods, let me give you a -- off the top of my head -- list of the different feature selection approaches

Filter methods:

Based on statistical tests measuring some intrinsic properties of the dataset / features

information gain

chi-square test

correlation coefficient

variance threshold

Wrapper methods:

recursive feature elimination

sequential feature selection algorithms

genetic algorithms

Embedded methods:

L1 (LASSO) regularization

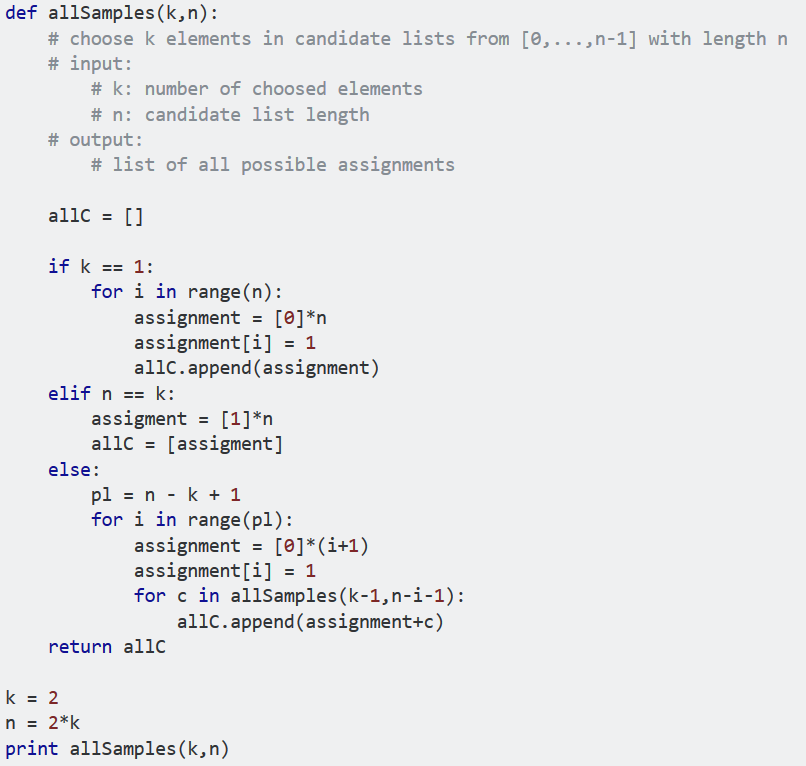
**Programming**

1. **Write a function to calculate all possible assignment vectors of 2n users, where n users are assigned to group 0 (control), and n users are assigned to group 1 (treatment).**

from itertools import combinations

def choices(n):

return list(combinations(range(2\*n), n))



5 You have a stream of data coming in of size n, but you don’t know what n is ahead of time. Write an algorithm that will take a random sample of k elements. Can you write one that takes O(k) space?

<https://en.wikipedia.org/wiki/Reservoir_sampling>

<https://www.quora.com/What-is-an-intuitive-explanation-of-reservoir-sampling>

6 Write an algorithm that can calculate the square root of a number.

<https://www.quora.com/What-is-the-method-to-calculate-a-square-root-by-hand?redirected_qid=664405>

8. When can parallelism make your algorithms run faster? When could it make your algorithms run slower?

It all depends on how well you can divide up your problem into individual parts that don't need to communicate. Image processing is a pretty good example where you in many cases can simply divide the image into parts and process them all individually in parallel and then put the parts together at the end.

The important part is that it works well because once you've divided the image into pieces, none of those pieces need information from any of the others while the processing takes place. In such a scenario you can simply put each piece on a separate processor.

When you run into problems is when you need to exchange information in order to continue executing. Then the process that needs information will just have to wait until that information is available. In such a scenario, you might be better off going single threaded.

But there's another reason for avoiding parallelism; it makes your program harder to understand and to reason about. Especially if it deals with a shared resource where you need to make sure that multiple threads don't step on each other's toes. This is why programming languages like Erlang, that specializes in parallel programming have a 'share nothing' policy

But the bottom line is that if you don't have a problem that's easily divided into chunks that don't need to communicate, then parallelism will not yield that many benefits.

**Statistical Inference**

1 In an A/B test, how can you check if assignment to the various buckets was truly random?

<https://www.quora.com/How-can-you-test-that-your-random-assignment-was-truly-random>

2 What might be the benefits of running an A/A test, where you have two buckets who are exposed to the exact same product?

<https://www.quora.com/What-is-an-A-A-test>

6 How would you run an A/B test for many variants, say 20 or more?

<https://www.quora.com/How-would-you-run-an-A-B-test-for-many-variants-say-20-or-more>

3 What is the curse of dimensionality?

<https://www.quora.com/What-is-the-curse-of-dimensionality>

4 Is more data always better?

<https://www.quora.com/Is-more-data-always-better>

5 What are advantages of plotting your data before performing analysis?

<https://www.quora.com/Why-do-economists-look-at-descriptive-statistics-and-plot-data-before-running-regressions>

6 How can you make sure that you don’t analyze something that ends up meaningless?

<https://www.quora.com/How-can-you-make-sure-that-you-dont-analyze-something-that-ends-up-meaningless>

9 How do you deal with some of your predictors being missing?

<https://www.quora.com/How-can-I-deal-with-missing-values-in-a-predictive-model>

12 Now you have a feasible amount of predictors, but you’re fairly sure that you don’t need all of them. How would you perform feature selection on the dataset?

<https://www.quora.com/How-do-I-perform-feature-selection>

15 What is the main idea behind ensemble learning? If I had many diﬀerent models that predicted the same response variable, what might I want to do to incorporate all of the models? Would you expect this to perform better than an individual model or worse?

<https://www.quora.com/What-are-the-best-methods-for-combining-different-machine-learning-models-to-get-a-better-prediction-than-any-individual-model>

<http://www.cnblogs.com/guolei/archive/2013/05/21/3091301.html>