

Attention

Все вопросы и ответы будут перемещены в

<https://github.com/iamtodor/data-science-interview-questions-and-answers> в течении некоторого, надеюсь, короткого времени.

Pull requests, замечания, комментарии, редактирования - все приветствуется. Вопросы и ответы не идеальны (еще).

Все содержание переместилось на страницу ниже.

Why do you use feature selection?

Explain what regularization is and why it is useful.

What's the difference between L1 and L2 regularization?

How would you validate a model you created to generate a predictive model of a quantitative outcome variable using multiple regression

Explain what precision and recall are. How do they relate to the ROC curve?

Is it better to have too many false positives, or too many false negatives? Explain.

How do you deal with unbalanced binary classification?

What is statistical power?

What is selection bias, why is it important and how can you avoid it?

What are bias and variance, and what are their relation to modeling data?

What if the classes are imbalanced? What if there are more than 2 groups?

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

12. In unsupervised learning, if a ground truth about a dataset is unknown, how can we determine the most useful number of clusters to be?

13. Define variance

14. Expected value

15. Describe the differences between and use cases for box plots and histograms

16. What features would you use to build a recommendation algorithm for users?

17. How would you find an anomaly in a distribution?

18. How would you go about investigating if a certain trend in a distribution is due to an anomaly?

19. Big Data Engineer Can you explain what REST is?

Logistic regression

What is the effect on the coefficients of logistic regression if two predictors are highly correlated? What are the confidence intervals of the coefficients?

What's the difference between Gaussian Mixture Model and K-Means?

Describe how Gradient Boosting works.

В чем разница между AdaBoost & XGBoost?

Data Mining Describe the decision tree model.

Data Mining What is a neural network?

How do you deal with sparse data?

Name and describe three different kernel functions and in what situation you would use each.

[Explain overfitting and how do you prevent overfitting?](#)

[How do you deal with outliers in your data?](#)

[What's the difference between supervised learning and unsupervised learning?](#)

[What is cross-validation and why would you use it?](#)

[What's the name of the matrix used to evaluate predictive models?](#)

[What's the relationship between Principal Component Analysis \(PCA\) and Linear & Quadratic Discriminant Analysis \(LDA & QDA\)](#)

[PCA vs SVD](#)

[If you had a categorical dependent variable and a mixture of categorical and continuous independent variables, what algorithms, methods, or tools would you use for analysis?](#)

[What's the difference between logistic and linear regression? How do you avoid local minima?](#)

[Explain a machine learning algorithm as if you're talking to a non-technical person.](#)

[How would you build a model to predict credit card fraud?](#)

[How do you handle missing or bad data?](#)

[If you're attempting to predict a customer's gender, and you only have 100 data points, what problems could arise?](#)

[Describe a non-normal probability distribution and how to apply it](#)

[Data Mining Explain what heteroskedasticity is and how to solve it](#)

[What are some different Time Series forecasting techniques?+](#)

[Explain Principle Component Analysis \(PCA\) and equations PCA uses.+](#)

[How do you solve Multicollinearity?+](#)

[What are p-values and confidence intervals?+](#)

[Data Analyst If you have 70 red marbles, and the ratio of green to red marbles is 2 to 7, how many green marbles are there?+](#)

[Given a die, would it be more likely to get a single 6 in six rolls, at least two 6s in twelve rolls, or at least one-hundred 6s in six-hundred rolls?](#)

[An SVM model is suffering with under fitting.](#)

[What's the Central Limit Theorem, and how do you prove it? What are its applications?](#)

[Функция распределения](#)

[Statistical Interactions](#)

[What error metric would you use to evaluate how good a binary classifier is?-](#)

[Assuming a clustering model's labels are known, how do you evaluate the performance of the model?-](#)

[Choose any machine learning algorithm and describe it.-](#)

[Describe a method used in machine learning.-](#)

[How would you derive new features from features that already exist?-](#)

[Suppose you were given two years of transaction history. What features would you use to predict credit risk?-](#)

[Explain Cross-validation as if you're talking to a non-technical person-](#)

Why do you use feature selection?

Feature selection is the process of selecting a subset of relevant features for use in model construction.

Feature selection is itself useful, but it mostly acts as a filter, muting out features that aren't useful in addition to your existing features.

Feature selection methods aid you in your mission to create an accurate predictive model. They help you by choosing features that will give you as good or better accuracy whilst requiring less data.

Feature selection methods can be used to identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model or may in fact decrease the accuracy of the model.

Fewer attributes is desirable because it reduces the complexity of the model, and a simpler model is simpler to understand and explain.

Filter Methods

Filter feature selection methods apply a statistical measure to assign a scoring to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods are often univariate and consider the feature independently, or with regard to the dependent variable. Some examples of some filter methods include the Chi squared test, information gain and correlation coefficient scores.

Embedded Methods

Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are regularization methods. Regularization methods are also called penalization methods that introduce additional constraints into the optimization of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (fewer coefficients).

Examples of regularization algorithms are the LASSO, Elastic Net and Ridge Regression.

Misleading

Including redundant attributes can be misleading to modeling algorithms. Instance-based methods such as k-nearest neighbor use small neighborhoods in the attribute space to determine classification and regression predictions. These predictions can be greatly skewed by redundant attributes.

Overfitting

Keeping irrelevant attributes in your dataset can result in overfitting. Decision tree algorithms like C4.5 seek to make optimal splits in attribute values. Those attributes that are more correlated with the prediction are split on first. Deeper in the tree less relevant and irrelevant attributes are used to make prediction decisions that may only be beneficial by chance in the training dataset. This overfitting of the training data can negatively affect the modeling power of the method and cripple the predictive accuracy.

Explain what regularization is and why it is useful.

Regularization is the process of adding a tuning parameter to a model to induce smoothness in order to prevent [overfitting](#).

This is most often done by adding a constant multiple to an existing weight vector. This constant is often either the [L1 \(Lasso\)](#) or [L2 \(ridge\)](#), but can in actuality can be any norm. The model predictions should then minimize the mean of the loss function calculated on the regularized training set.

It is well known, as explained by others, that L1 regularization helps perform feature selection in sparse feature spaces, and that is a good practical reason to use L1 in some situations. However, beyond that particular reason I have never seen L1 to perform better than L2 in practice. If you take a look at [LIBLINEAR FAQ](#) on this issue you will see how they have not seen a practical example where L1 beats L2 and encourage users of the library to contact them if they find one. Even in a situation where you might benefit from L1's sparsity in order to do feature selection, using L2 on the remaining variables is likely to give better results than L1 by itself.

What's the difference between L1 and L2 regularization?

Regularization is a very important technique in machine learning to prevent overfitting. Mathematically speaking, it adds a *regularization term* in order to prevent the coefficients to fit so perfectly to overfit. The difference between the L1(Lasso) and L2(Ridge) is just that L2(Ridge) is the sum of the square of the weights, while L1(Lasso) is just the sum of the absolute weights in MSE or another loss function. As follows:

L1 regularization on least squares:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^k |w_i|$$

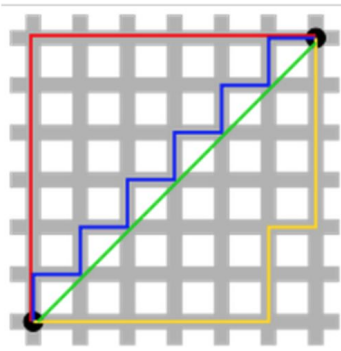
L2 regularization on least squares:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^k w_i^2$$

The difference between their properties can be promptly summarized as follows:

L2 regularization	L1 regularization
Computational efficient due to having analytical solutions	Computational inefficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No feature selection	Built-in feature selection

Solution uniqueness is a simpler case but requires a bit of imagination. First, this picture below:



The green line (L2-norm) is the unique shortest path, while the red, blue, yellow (L1-norm) are all same length (=12) for the same route. Generalizing this to n-dimensions. This is why L2-norm has unique solutions while L1-norm does not.

Built-in feature selection is frequently mentioned as a useful property of the L1-norm, which the L2-norm does not. This is actually a result of the L1-norm, which tends to produce sparse coefficients (explained below). Suppose the model has 100 coefficients but only 10 of them have non-zero coefficients, this is effectively saying that “the other 90 predictors are useless in predicting the target values”. L2-norm produces non-sparse coefficients, so does not have this property.

Sparsity refers to that only very few entries in a matrix (or vector) is non-zero. L1-norm has the property of producing many coefficients with zero values or very small values with few large coefficients.

Computational efficiency. L1-norm does not have an analytical solution, but L2-norm does. This allows the L2-norm solutions to be calculated computationally efficiently. However, L1-norm solutions do have the sparsity properties which allows it to be used along with sparse algorithms, which makes the calculation more computationally efficient.

How would you validate a model you created to generate a predictive model of a quantitative outcome variable using multiple regression

Proposed methods for model validation:

- If the values predicted by the model are far outside of the response variable range, this would immediately indicate poor estimation or model inaccuracy.
- If the values seem to be reasonable, examine the parameters; any of the following would indicate poor estimation or multi-collinearity: opposite signs of expectations, unusually large or small values, or observed inconsistency when the model is fed new data.
- Use the model for prediction by feeding it new data, and use the coefficient of determination (R squared) as a model validity measure.
- Use data splitting to form a separate dataset for estimating model parameters, and another for validating predictions.
- Use jackknife resampling if the dataset contains a small number of instances, and measure validity with R squared and mean squared error (MSE).

Explain what precision and recall are. How do they relate to the ROC curve?

Calculating precision and recall is actually quite easy. Imagine there are 100 positive cases among 10,000 cases. You want to predict which ones are positive, and you pick 200 to have a better chance of catching many of the 100 positive cases. You record the IDs of your predictions, and when you get the actual results you sum up how many times you were right or wrong. There are four ways of being right or wrong:

1. TN / True Negative: case was negative and predicted negative
2. TP / True Positive: case was positive and predicted positive
3. FN / False Negative: case was positive but predicted negative
4. FP / False Positive: case was negative but predicted positive

	Predicted Negative	Predicted Positive
Negative Cases	TN: 9,760	FP: 140
Positive Cases	FN: 40	TP: 60

Now, your boss asks you three questions:

1. What percent of your predictions were correct?

You answer: the "accuracy" was (9,760+60) out of 10,000 = 98.2%

2. What percent of the positive cases did you catch?

You answer: the "recall" was 60 out of 100 = 60%

3. What percent of positive predictions were correct?

You answer: the "precision" was 60 out of 200 = 30%

See also a very good explanation of [Precision and recall](#) in Wikipedia.

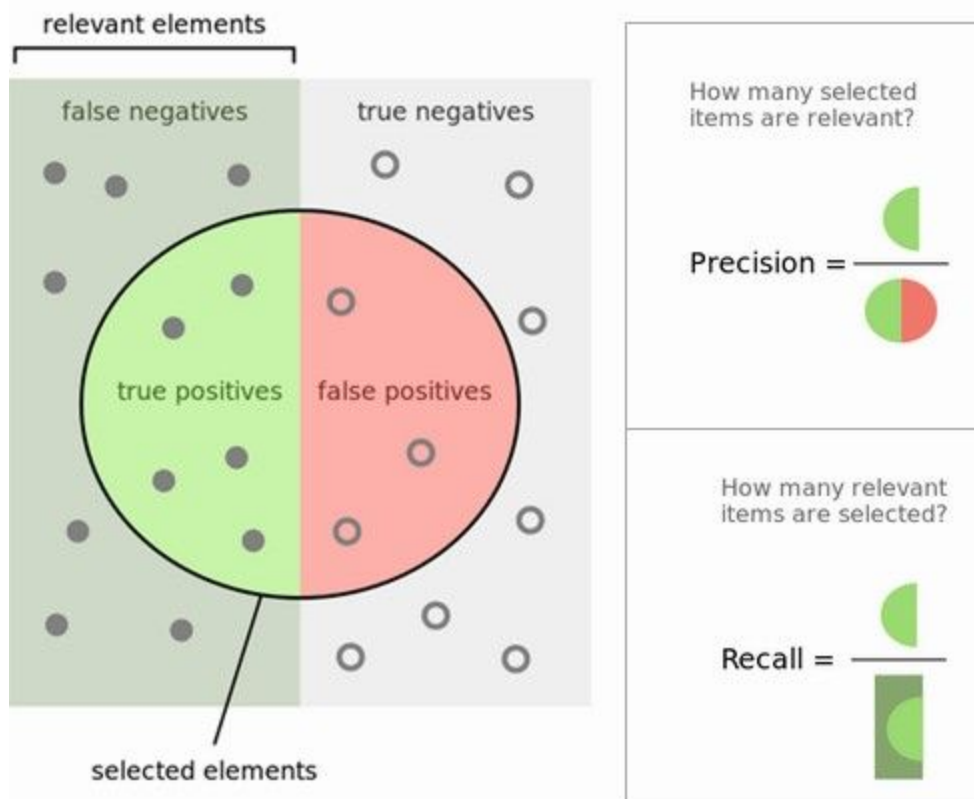


Fig 4: Precision and Recall.

ROC curve represents a relation between sensitivity (RECALL) and specificity (NOT PRECISION) and is commonly used to measure the performance of binary classifiers. However, when dealing with highly skewed datasets, [Precision-Recall \(PR\) curves](#) give a more representative picture of performance. Remember, a ROC curve represents a relation between sensitivity (RECALL) and specificity (NOT PRECISION). Sensitivity is the other name for recall but specificity is not PRECISION.

Recall/Sensitivity is the measure of the probability that your estimate is 1 given all the samples whose true class label is 1. It is a measure of how many of the positive samples have been identified as being positive. Specificity is the measure of the probability that your estimate is 0 given all the samples whose true class label is 0. It is a measure of how many of the negative samples have been identified as being negative.

PRECISION on the other hand is different. It is a measure of the probability that a sample is a true positive class given that your classifier said it is positive. It is a measure of how many of the samples predicted by the classifier as positive is indeed positive. Note here that this changes when the base probability or prior probability of the positive class changes. Which means PRECISION depends on how rare is the positive class. In other words, it is used when positive class is more interesting than the negative class.

- Sensitivity also known as the True Positive rate or Recall is calculated as,

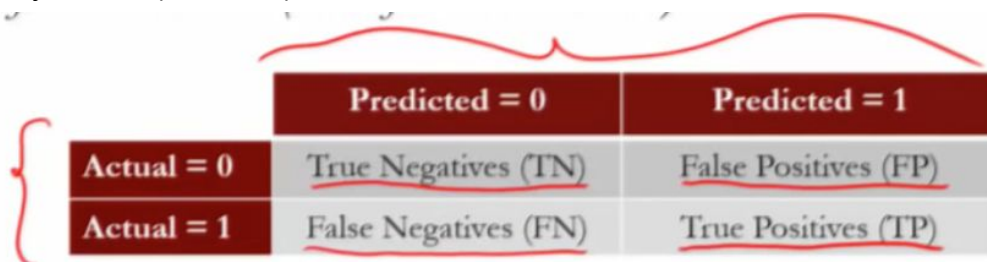
$$\text{Sensitivity} = \text{TP} / (\text{TP} + \text{FN})$$

Since the formula doesn't contain FP and TN, Sensitivity may give you a biased result, especially for imbalanced classes.

In the example of Fraud detection, it gives you the percentage of Correctly Predicted Frauds from the pool of Actual Frauds.

- Specificity, also known as True Negative Rate is calculated as,

$$\text{Specificity} = \text{TN} / (\text{TN} + \text{FP})$$



A confusion matrix diagram with red annotations. A large red curly brace on the left groups the two rows, labeled 'Actual = 0' and 'Actual = 1'. A red bracket at the top groups the two columns, labeled 'Predicted = 0' and 'Predicted = 1'. The matrix cells contain: True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP). Each cell text is underlined in red.

	Predicted = 0	Predicted = 1
Actual = 0	<u>True Negatives (TN)</u>	<u>False Positives (FP)</u>
Actual = 1	<u>False Negatives (FN)</u>	<u>True Positives (TP)</u>

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Since the formula does not contain FN and TP, Specificity may give you a biased result, especially for imbalanced classes.

In the example of Fraud detection, it gives you the percentage of Correctly Predicted Non-Frauds from the <http://machinelearningmastery.com/assessing-comparing-classifier-performance-roc-curves-2/> <https://alexanderdyakonov.wordpress.com/2017/07/28/auc-roc-%D0%BF%D0%BB%D0%BE%D1%89%D0%B0%D0%B4%D1%8C-%D0%BF%D0%BE%D0%B4-%D0%BA%D1%80%D0%B8%D0%B2%D0%BE%D0%B9-%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA/>

Is it better to have too many false positives, or too many false negatives?

Explain.

It depends on the question as well as on the domain for which we are trying to solve the question.

In medical testing, false negatives may provide a falsely reassuring message to patients and physicians that disease is absent, when it is actually present. This sometimes leads to inappropriate or inadequate treatment of both the patient and their disease. So, it is desired to have too many false positive.

For spam filtering, a false positive occurs when spam filtering or spam blocking techniques wrongly classify a legitimate email message as spam and, as a result, interferes with its delivery. While most anti-spam tactics can block or filter a high percentage of unwanted emails, doing so without creating significant false-positive results is a much more demanding task. So, we prefer too many false negatives over many false positives.

How do you deal with unbalanced binary classification?

Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally.

For example, you may have a 2-class (binary) classification problem with 100 instances (rows). A total of 80 instances are labeled with Class-1 and the remaining 20 instances are labeled with Class-2.

This is an imbalanced dataset and the ratio of Class-1 to Class-2 instances is 80:20 or more concisely 4:1.

You can have a class imbalance problem on two-class classification problems as well as multi-class classification problems. Most techniques can be used on either.

The remaining discussions will assume a two-class classification problem because it is easier to think about and describe.

1) Can You Collect More Data?

A larger dataset might expose a different and perhaps more balanced perspective on the classes.

More examples of minor classes may be useful later when we look at resampling your dataset.

2) Try Changing Your Performance Metric

Accuracy is not the metric to use when working with an imbalanced dataset. We have seen that it is misleading.

From that post, I recommend looking at the following performance measures that can give more insight into the accuracy of the model than traditional classification accuracy:

- Confusion Matrix: A breakdown of predictions into a table showing correct predictions (the diagonal) and the types of incorrect predictions made (what classes incorrect predictions were assigned).
- Precision: A measure of a classifiers exactness. [Precision](#) is the number of True Positives divided by the number of True Positives and False Positives. Put another way, it is the number of positive predictions divided by the total number of positive class values predicted. It is also called the [Positive Predictive Value](#) (PPV). Precision can be thought of as a measure of a classifiers exactness. A low precision can also indicate a large number of False Positives.
- Recall: A measure of a classifiers completeness. [Recall](#) is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate. Recall can be thought of as a measure of a classifiers completeness. A low recall indicates many False Negatives.
- F1 Score (or F-score): A weighted average of precision and recall.

I would also advise you to take a look at the following:

- Kappa (or [Cohen's kappa](#)): Classification accuracy normalized by the imbalance of the classes in the data.
- ROC Curves: Like precision and recall, accuracy is divided into sensitivity and specificity and models can be chosen based on the balance thresholds of these values.

3) Try Resampling Your Dataset

1. You can add copies of instances from the under-represented class called over-sampling (or more formally sampling with replacement)
2. You can delete instances from the over-represented class, called under-sampling.

5) Try Different Algorithms

6) Try Penalized Models

You can use the same algorithms but give them a different perspective on the problem.

Penalized classification imposes an additional cost on the model for making classification mistakes on the minority class during training. These penalties can bias the model to pay more attention to the minority class.

Often the handling of class penalties or weights are specialized to the learning algorithm. There are penalized versions of algorithms such as penalized-SVM and penalized-LDA.

Using penalization is desirable if you are locked into a specific algorithm and are unable to resample or you're getting poor results. It provides yet another way to "balance" the classes. Setting up the penalty matrix can be complex. You will very likely have to try a variety of penalty schemes and see what works best for your problem.

7) Try a Different Perspective

Taking a look and thinking about your problem from these perspectives can sometimes shake loose some ideas.

Two you might like to consider are anomaly detection and change detection.

What is statistical power?

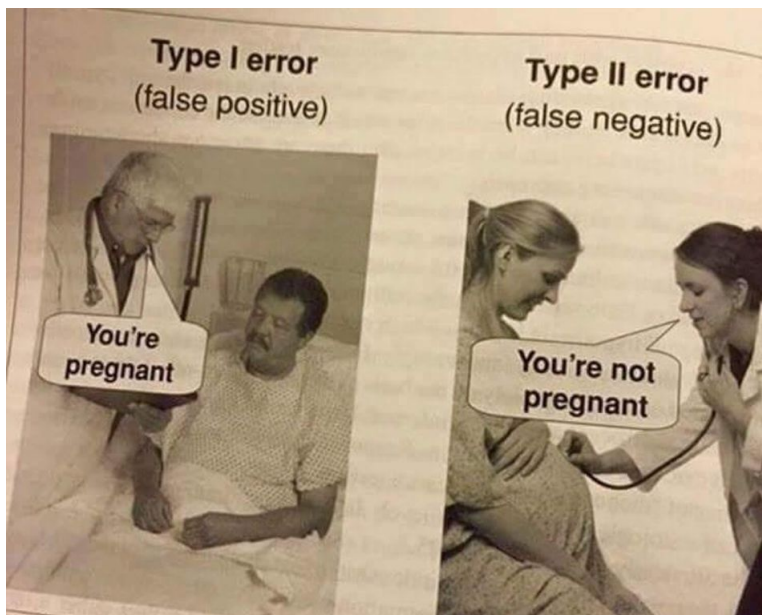
[Statistical power or sensitivity](#) of a binary hypothesis test is the probability that the test correctly rejects the null hypothesis (H_0) when the alternative hypothesis (H_1) is true.

It can be equivalently thought of as the probability of accepting the alternative hypothesis (H_1) when it is true—that is, the ability of a test to detect an effect, if the effect actually exists.

To put in another way, [Statistical power](#) is the likelihood that a study will detect an effect when the effect is present. The higher the statistical power, the less likely you are to make a Type II error (concluding there is no effect when, in fact, there is).

A type I error (or error of the first kind) is the incorrect rejection of a true null hypothesis. Usually a type I error leads one to conclude that a supposed effect or relationship exists when in fact it doesn't. Examples of type I errors include a test that shows a patient to have a disease when in fact the patient does not have the disease, a fire alarm going on indicating a fire when in fact there is no fire, or an experiment indicating that a medical treatment should cure a disease when in fact it does not.

A type II error (or error of the second kind) is the failure to reject a false null hypothesis. Examples of type II errors would be a blood test failing to detect the disease it was designed to detect, in a patient who really has the disease; a fire breaking out and the fire alarm does not ring; or a clinical trial of a medical treatment failing to show that the treatment works when really it does.



What is selection bias, why is it important and how can you avoid it?

Selection bias, in general, is a problematic situation in which error is introduced due to a non-random population sample. For example, if a given sample of 100 test cases was made up of a 60/20/15/5 split of 4 classes which actually occurred in relatively equal numbers in the population, then a given model may make the false assumption that probability could be the determining predictive factor. Avoiding non-random samples is the best way to deal with bias; however, when this is impractical, techniques such as [resampling](#), [boosting](#), and weighting are strategies which can be introduced to help deal with the situation.

What are bias and variance, and what are their relation to modeling data?

Bias is how far removed a model's predictions are from correctness, while **variance** is the degree to which these predictions vary between model iterations.

Error due to Bias: Due to randomness in the underlying data sets, the resulting models will have a range of predictions. Bias measures how far off in general these models' predictions are from the correct value. The [bias](#) is error from erroneous assumptions in the learning [algorithm](#). High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

Error due to Variance: The error due to variance is taken as the variability of a model prediction for a given data point. Again, imagine you can repeat the entire model building process multiple times. The variance is how much the predictions for a given point vary between different realizations of the model. The [variance](#) is error from sensitivity to small fluctuations in the training set.

High variance can cause an algorithm to model the random [noise](#) in the training data, rather than the intended outputs ([overfitting](#)).

Большой датасет -> низкий variance

Маленький датасет -> высокий variance

Мало фичей -> высокий bias, низкий variance

Много фичей -> низкий bias, высокий variance

Сложнее модель -> низкий bias

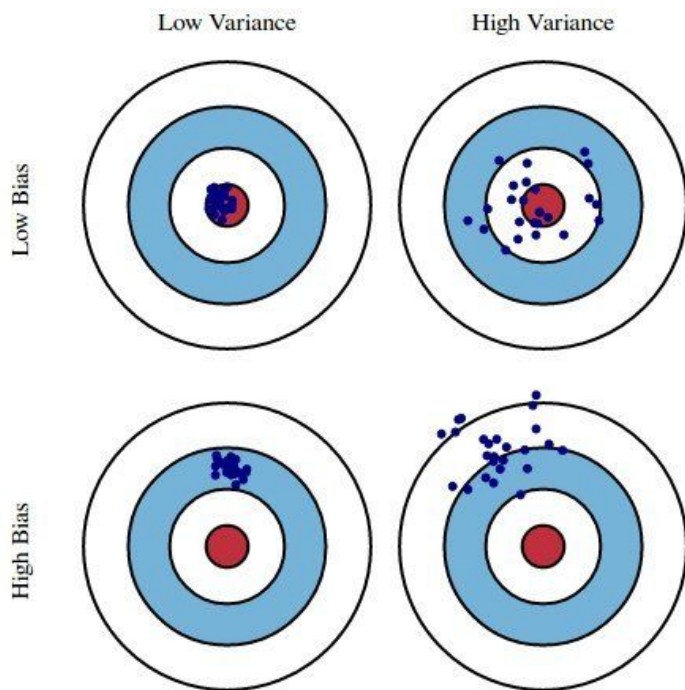
Проще модель -> высокий bias

Decreasing λ -> низкий bias

Increasing λ -> низкий variance.

Bias - это, условно говоря, расстояние между моделью которую ты можешь зафитить на бесконечных тренировочных данных (наилучшей моделью, которую может предоставить твоё пространство моделей) и "настоящей моделью" (которая генерирует данные).

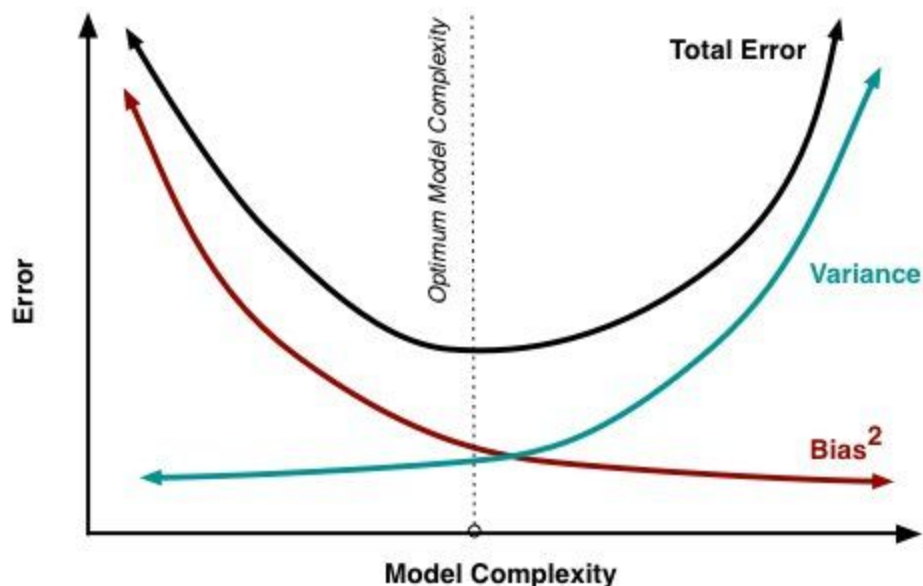
We can create a graphical visualization of bias and variance using a bulls-eye diagram. Imagine that the center of the target is a model that perfectly predicts the correct values. As we move away from the bulls-eye, our predictions get worse and worse. Imagine we can repeat our entire model building process to get a number of separate hits on the target. Each hit represents an individual realization of our model, given the chance variability in the training data we gather. Sometimes we will get a good distribution of training data so we predict very well and we are close to the bulls-eye, while sometimes our training data might be full of outliers or non-standard values resulting in poorer predictions. These different realizations result in a scatter of hits on the target.



As an example, using a simple flawed Presidential election survey as an example, errors in the survey are then explained through the twin lenses of bias and variance: selecting survey participants from a phonebook is a source of bias; a small sample size is a source of variance.

Minimizing total model error relies on the balancing of bias and variance errors. Ideally, models are the result of a collection of **unbiased data of low variance**. Unfortunately, however, the more complex a model becomes, its tendency is toward less bias but greater variance; therefore an optimal model would need to consider a balance between these 2 properties.

The statistical evaluation method of cross-validation is useful in both demonstrating the **importance** of this balance, as well as actually **searching** it out. The number of data folds to use -- the value of k in k -fold cross-validation -- is an important decision; the lower the value, the higher the bias in the error estimates and the less variance.



Bias and variance contributing to total error, [Image source](#). Conversely, when k is set equal to the number of instances, the error estimate is then very low in bias but has the possibility of high variance. The most important takeaways are that bias and variance are two sides of an important trade-off when building models, and that even the most routine of statistical evaluation methods are directly reliant upon such a trade-off.

We may estimate a model $\hat{f}(X)$ of $f(X)$ using linear regressions or another modeling technique. In this case, the expected squared prediction error at a point x is:

$$\text{Err}(x) = E[(Y - \hat{f}(x))^2]$$

This error may then be decomposed into bias and variance components:

$$\text{Err}(x) = (E[\hat{f}(x)] - f(x))^2 + E[(\hat{f}(x) - E[\hat{f}(x)])^2] + \sigma^2$$

$$\text{Err}(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible}$$

That third term, irreducible error, is the noise term in the true relationship that cannot fundamentally be reduced by any model. Given the true model and infinite data to calibrate it, we should be able to reduce both the bias and variance terms to 0. However, in a world with imperfect models and finite data, there is a tradeoff between minimizing the bias and minimizing the variance.

If a model is suffering from high bias, it means that model is less complex, to make the model more robust, we can add more features in feature space. Adding data points will reduce the variance.

The bias–variance tradeoff is a central problem in supervised learning. Ideally, one wants to [choose a model](#) that both accurately captures the regularities in its training data, but also [generalizes](#) well to unseen data. Unfortunately, it is typically impossible to do both simultaneously. High-variance learning methods may be able to represent their training set well, but are at risk of overfitting to noisy or unrepresentative training data. In contrast, algorithms with high bias typically produce simpler models that don't tend to overfit, but may *underfit* their training data, failing to capture important regularities.

Models with low bias are usually more complex (e.g. higher-order regression polynomials), enabling them to represent the training set more accurately. In the process, however, they may also represent a large [noise](#) component in the training set, making their predictions less accurate - despite their added complexity. In contrast, models with higher bias tend to be relatively simple (low-order or even linear regression polynomials), but may produce lower variance predictions when applied beyond the training set.

Approaches

[Dimensionality reduction](#) and [feature selection](#) can decrease variance by simplifying models. Similarly, a larger training set tends to decrease variance. Adding features (predictors) tends to decrease bias, at the expense of introducing additional variance. Learning algorithms typically have some tunable parameters that control bias and variance, e.g.:

- ([Generalized](#)) linear models can be [regularized](#) to decrease their variance at the cost of increasing their bias.
- In [artificial neural networks](#), the variance increases and the bias decreases with the number of hidden units. Like in GLMs, regularization is typically applied.
- In [k-nearest neighbor](#) models, a high value of k leads to high bias and low variance (see below).
- In [Instance-based learning](#), regularization can be achieved varying the mixture of [prototypes](#) and exemplars.[†]
- In [decision trees](#), the depth of the tree determines the variance. Decision trees are commonly pruned to control variance.

One way of resolving the trade-off is to use [mixture models](#) and [ensemble learning](#). For example, [boosting](#) combines many "weak" (high bias) models in an ensemble that has lower bias than the individual models, while [bagging](#) combines "strong" learners in a way that reduces their variance.

What if the classes are imbalanced? What if there are more than 2 groups?

Binary classification involves classifying the data into two groups, e.g. whether or not a customer buys a particular product or not (Yes/No), based on independent variables such as gender, age, location etc.

As the target variable is not continuous, binary classification model predicts the probability of a target variable to be Yes/No. To evaluate such a model, a metric called the confusion matrix is used, also called the classification or co-incidence matrix. With the help of a confusion matrix, we can calculate important performance measures:

- True Positive Rate (TPR) or Hit Rate or Recall or Sensitivity = $TP / (TP + FN)$
- Precision = $TP / (TP + FP)$
- False Positive Rate(FPR) or False Alarm Rate = $1 - \text{Specificity} = 1 - (TN / (TN + FP))$
- Accuracy = $(TP + TN) / (TP + TN + FP + FN)$
- Error Rate = $1 - \text{accuracy}$ or $(FP + FN) / (TP + TN + FP + FN)$
- F-measure: $2 / ((1 / \text{Precision}) + (1 / \text{Recall}))$
- ROC (Receiver Operating Characteristics) = plot of FPR vs TPR
- AUC (Area Under the Curve)

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

There are several ways to make a model more robust to outliers, from different points of view (data preparation or model building). **An outlier** in the question and answer is assumed being unwanted,

unexpected, or a must-be-wrong value to the human's knowledge so far (e.g. no one is 200 years old) rather than a rare event which is possible but rare.

Outliers are usually defined in relation to the distribution. Thus outliers could be removed in the pre-processing step (before any learning step), by using standard deviations (for normality) or interquartile ranges (for not normal/unknown) as threshold levels.

Moreover, **data transformation** (e.g. log transformation) may help if data have a noticeable tail. When outliers related to the sensitivity of the collecting instrument which may not precisely record small values, **Winsorization** may be useful. This type of transformation (named after Charles P. Winsor (1895–1951)) has the same effect as clipping signals (i.e. replaces extreme data values with less extreme values). Another option to reduce the influence of outliers is using **mean absolute difference** rather than mean squared error.

For model building, some models are resistant to outliers (e.g. [tree-based approaches](#)) or non-parametric tests. Similar to the median effect, tree models divide each node into two in each split. Thus, at each split, all data points in a bucket could be equally treated regardless of extreme values they may have.

12. In unsupervised learning, if a ground truth about a dataset is unknown, how can we determine the most useful number of clusters to be?

The elbow method is often the best place to state, and is especially useful due to its ease of explanation and verification via visualization. The elbow method is interested in explaining variance as a function of cluster numbers (the k in k -means). By plotting the percentage of variance explained against k , the first N clusters should add significant information, explaining variance; yet, some eventual value of k will result in a much less significant gain in information, and it is at this point that the graph will provide a noticeable angle. This angle will be the optimal number of clusters, from the perspective of the elbow method,

It should be self-evident that, in order to plot this variance against varying numbers of clusters, varying numbers of clusters must be tested. Successive complete iterations of the clustering method must be undertaken, after which the results can be plotted and compared.

DBSCAN - Density-Based Spatial Clustering of Applications with Noise. Finds core samples of high density and expands clusters from them. Good for data which contains clusters of similar density.

13. Define variance

Variance is the [expectation](#) of the squared [deviation](#) of a [random variable](#) from its [mean](#), мера разброса значений [случайной величины](#) относительно её [математического ожидания](#). Informally, it measures how far a set of (random) numbers are spread out from their average value. The variance is the square of the [standard deviation](#), the second [central moment](#) of a distribution, and the [covariance](#) of the random variable with itself.

$$\text{Var}(X) = E[(X - m)^2], m = E[X]$$

14. Expected value

Математическое ожидание — среднее значение случайной величины (распределение вероятностей случайной величины, рассматривается в теории вероятностей). Значение, которое случайная величина принимает с наибольшей вероятностью.

Предположим теперь, что мы знаем закон распределения случайной величины x , то есть знаем, что случайная величина x может принимать значения x_1, x_2, \dots, x_k с вероятностями p_1, p_2, \dots, p_k .

Математическое ожидание M_x случайной величины x равно.

Математическое ожидание случайной величины X (обозначается $M(X)$ или реже $E(X)$) характеризует среднее значение случайной величины (дискретной или непрерывной). Мат. ожидание - это первый начальный момент заданной СВ.

Математическое ожидание относят к так называемым характеристикам положения распределения (к которым также принадлежат мода и медиана). Эта характеристика описывает некое усредненное положение случайной величины на числовой оси. Скажем, если математическое ожидание случайной величины - срока службы лампы, равно 100 часов, то считается, что значения срока службы сосредоточены (с обеих сторон) от этого значения (с тем или иным разбросом, о котором уже говорит дисперсия).

Математическое ожидание дискретной случайной величины X вычисляется как сумма произведений значений x_i , которые принимает СВ X , на соответствующие вероятности p_i :

$$M(X) = \sum_{i=1}^n x_i \cdot p_i.$$

Для **непрерывной случайной величины** (заданной плотностью вероятностей $f(x)$), формула вычисления математического ожидания X выглядит следующим образом:

$$M(X) = \int_{-\infty}^{+\infty} f(x) \cdot x dx.$$

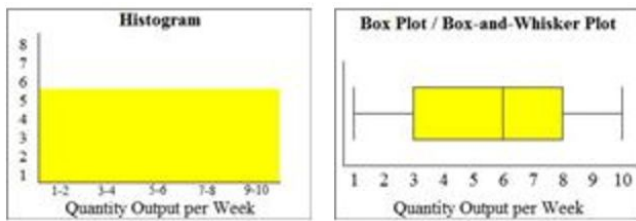
15. Describe the differences between and use cases for box plots and histograms

A [histogram](#) is a type of bar chart that graphically displays the frequencies of a data set. Similar to a bar chart, a histogram plots the frequency, or raw count, on the Y-axis (vertical) and the variable being measured on the X-axis (horizontal).

The only difference between a histogram and a bar chart is that a histogram displays frequencies for a group of data, rather than an individual data point; therefore, no spaces are present between the bars. Typically, a histogram groups data into small chunks (four to eight values per bar on the horizontal axis), unless the range of data is so great that it is easier to identify general distribution trends with larger groupings.

A box plot, also called a [box-and-whisker plot](#), is a chart that graphically represents the five most important descriptive values for a data set. These values include the minimum value, the first quartile, the median, the third quartile, and the maximum value. When graphing this five-number summary, only the horizontal axis displays values. Within the quadrant, a vertical line is placed above each of the summary numbers. A box is drawn around the middle three lines (first quartile, median, and third quartile) and two lines are drawn from the box's edges to the two endpoints (minimum and maximum).

Boxplots are better for comparing distributions than histograms!



16. What features would you use to build a recommendation algorithm for users?

17. How would you find an anomaly in a distribution?

Best steps to prevent anomalies is to implement policies or checks that can catch them during the data collection stage. Unfortunately, you do not often get to collect your own data, and often the data you're mining was collected for another purpose. About 68% of all the data points are within one standard deviation from the mean. About 95% of the data points are within two standard deviations from the mean. Finally, over 99% of the data is within three standard deviations from the mean. When the value deviate too much from the mean, let's say by $\pm 4\sigma$, then we can considerate this almost impossible value as anomaly. (This limit can also be calculated using the percentile).

Statistical methods

Statistically based anomaly detection uses this knowledge to discover outliers. A dataset can be standardized by taking the z-score of each point. A z-score is a measure of how many standard deviations a data point is away from the mean of the data. Any data-point that has a z-score higher than 3 is an outlier, and likely to be an anomaly. As the z-score increases above 3, points become more obviously anomalous. A z-score is calculated using the following equation. A box-plot is perfect for this application.

Итерационные методы

Методы, которые состоят из итераций, на каждой из которых удаляется группа «особо подозрительных объектов». Например, в n-мерном признаковом пространстве можно удалять выпуклую оболочку наших точек-объектов, считая её представителей выбросами. Как правило, методы этой группы достаточно трудоёмким.

Метрические методы

Судя по числу публикаций, это самые популярные методы среди исследователей. В них постулируется существование некоторой метрики в пространстве объектов, которая и помогает найти аномалии. Интуитивно понятно, что у выброса мало соседей, а у типичной точки много. Поэтому хорошей мерой аномальности может служить, например «расстояние до k-го соседа» (см. метод [Local Outlier Factor](#)). Здесь используются специфические метрики, например [расстояние Махаланобиса](#). Мера [расстояния](#) между векторами случайных величин, обобщающая понятие евклидова расстояния. С помощью расстояния Махаланобиса можно определять *сходство* неизвестной и известной [выборки](#). Оно отличается от [расстояния Евклида](#) тем, что учитывает [корреляции](#) между переменными и инвариантно к масштабу.

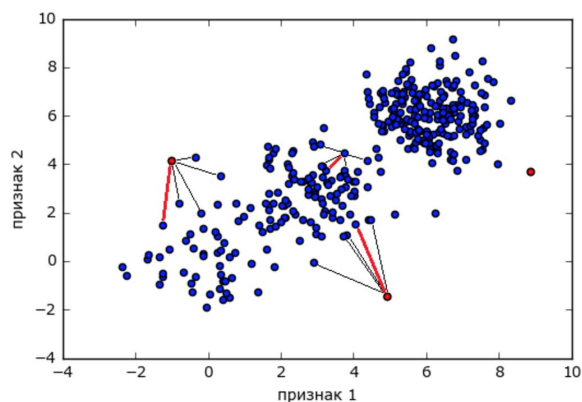


Рис. 8. Соседи нескольких элементов выборки, связь с 5м показана красным

Методы подмены задачи

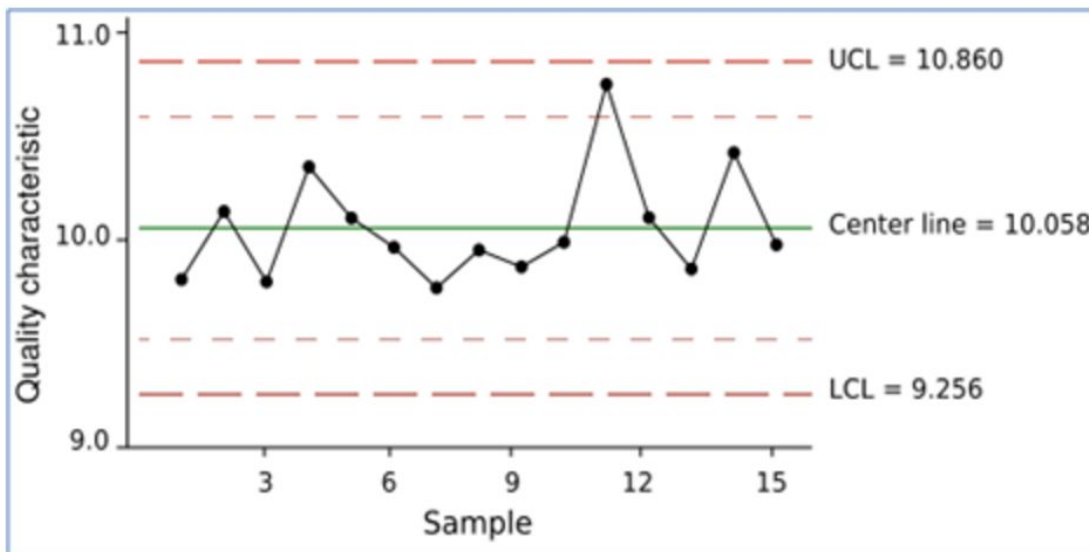
Когда возникает новая задача, есть большой соблазн решить её старыми методами (ориентированными на уже известные задачи). Например, можно сделать кластеризацию, тогда маленькие кластеры, скорее всего, состоят из аномалий. Если у нас есть частичная информация об аномалиях (как в задаче PUC), то можно решить её как задачу классификации с классами 1 (размеченные аномалии) и 0 (все остальные объекты). Если бы класс 0 состоял только из нормальных объектов, то такое решение было бы совсем законным, иначе остаётся надеяться, что недетектированных аномалий в нём немного.

The most common form of clustering-based anomaly detection is done with prototype-based clustering. Using this approach to anomaly detection, a point is classified as an anomaly if its omission from the group significantly improves the prototype, then the point is classified as an anomaly. This logically makes sense. K-means is a clustering algorithm that clusters similar points. The points in any cluster are similar to the centroid of that cluster, hence why they are members of that cluster. If one point in the cluster is so far from the centroid that it pulls the centroid away from its natural center, then that point is literally an outlier, since it lies outside the natural bounds for the cluster. Hence, its omission is a logical step to improve the accuracy of the rest of the cluster. Using this approach, the outlier score is defined as the degree to which a point doesn't belong to any cluster, or the distance it is from the centroid of the cluster. In K-means, the degree to which the removal of a point would increase the accuracy of the centroid is the difference in the SSE, or standard squared error, or the cluster with and without the point. If there is a substantial improvement in SSE after the removal of the point, that correlates to a high outlier score for that point. More specifically, when using a k-means clustering approach towards anomaly detection, the outlier score is calculated in one of two ways. The simplest is the point's distance from its closest centroid. However, this approach is not as useful when there are clusters of differing densities. To tackle that problem, the point's relative distance to its closest centroid is used, where relative distance is defined as the ratio of the point's distance from the centroid to the median distance of all points in the cluster from the centroid. This approach to anomaly detection is sensitive to the value of k. Also, if the data is highly noisy, then that will throw off the accuracy of the initial clusters, which will decrease the accuracy of this type of anomaly detection. The time complexity of this approach is obviously dependent on the choice of clustering algorithm, but since most clustering algorithms have linear or close to linear time and space complexity, this type of anomaly detection can be highly efficient.

18. How would you go about investigating if a certain trend in a distribution is due to an anomaly?

When you estimate the model you'll get the variance σ_{ϵ} of the error ϵ_t . Depending on your distributional assumptions, such as normal, you can set the threshold based on the probability, such as $|\epsilon_t| < 3\sigma_{\epsilon}$ for 99.7% or one-sided $\epsilon_t > 3\sigma_{\epsilon}$.

- **Moving Averages:** In statistics, a moving average can be used to analyze a set of data points by creating a series of averages of different time period subsets (e.g., 6 week moving average, 3 month moving average) of the full data set. A moving average can be used to identify processes or situations that are trending significantly up or down.
- **Comparative Analysis (Previous Period, Previous Campaign or Event, Benchmarks):** Comparing current performance to previous periods or previous campaigns (e.g., Back To School Campaign) or previous events (e.g., elections) or industry benchmarks is an easy way to flag areas of under- and over-performance (see chart below).
- **Control Charts:** Control charts are used to determine whether a process is in a state of statistical control. Control charts indicate upper and lower control limits, and include a central (average) line, to help detect trend of plotted values. If all data points are within the control limits, variations in the values may be due to a common cause and process is said to be 'in control'. If data points fall outside the control limits, variations may be due to a special cause and the process is said to be out of control.



- **Basic Statistical Analysis:** Calculating average (mean) and standard deviations are standard calculations available in most BI tools. You can use basic statistical analysis to flag anomalies that are 3 standard deviations (as a measure of spread) from the average or mean (the center of the distribution).

Twitter's AnomalyDetection Package

- Works by using Seasonal Hybrid ESD (S-H-ESD);
- Builds upon the Generalized ESD test for detecting anomalies;
- Can detect both local and global anomalies;
- Employing time series decomposition and robust statistical metrics (e.g. median together with ESD)
- Employs piecewise approximation for long time series;
- Also has method for when time stamps are not available;
- Can specify direction of anomalies, window of interest, toggle the piecewise approximation, and has visuals support.

If $p(x) < E$, p - probability, E - epsilon

19. Big Data Engineer Can you explain what REST is?

REST stands for Representational State Transfer. (It is sometimes spelled "ReST".) It relies on a stateless, client-server, cacheable communications protocol -- and in virtually all cases, the HTTP protocol is used. REST is an architecture style for designing networked applications. The idea is simple HTTP is used to make calls between machines.

- In many ways, the World Wide Web itself, based on HTTP, can be viewed as a REST-based architecture.

RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations.

REST is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP, WSDL, et al.). Later, we will see how much more simple REST is.

- Despite being simple, REST is fully-featured; there's basically nothing you can do in Web Services that can't be done with a RESTful architecture.

REST is not a "standard". There will never be a W3C recommendation for REST, for example. And while there are REST programming frameworks, working with REST is so simple that you can often "roll your own" with standard library features in languages like Perl, Java, or C#.

Logistic regression

Log odds - raw output from the model; odds - exponent from the output of the model. Probability of the output - odds / (1+odds).

What is the effect on the coefficients of logistic regression if two predictors are highly correlated? What are the confidence intervals of the coefficients?

When predictor variables are correlated, the estimated regression coefficient of any one variable depends on which other predictor variables are included in the model. When predictor variables are correlated, the precision of the estimated regression coefficients decreases as more predictor variables are added to the model.

In [statistics](#), multicollinearity (also collinearity) is a phenomenon in which two or more predictor [variables](#) in a [multiple regression](#) model are highly [correlated](#), meaning that one can be linearly predicted from the others with a substantial degree of accuracy. In this situation the [coefficient estimates](#) of the multiple regression may change erratically in response to small changes in the model or the data. Multicollinearity does not reduce the predictive power or reliability of the model as a whole, at least within the sample data set; it only affects calculations regarding [individual predictors](#). That is, a multiple regression model with correlated predictors can indicate how well the entire bundle of predictors predicts the [outcome variable](#), but it may not give valid results about any individual predictor, or about which predictors are redundant with respect to others.

Последствия мультиколлинеарности:

Оценки коэффициентов остаются несмещенными

Стандартные ошибки коэффициентов увеличиваются

Вычисленные t-статистики занижены.

Оценки становятся очень чувствительными к изменению спецификации и изменению отдельных наблюдений.

Общее качество уравнения, а также оценки переменных, не связанных мультиколлинеарностью, остаются незатронутыми.

Чем ближе мультиколлинеарность к совершенной (строгой), тем серьезнее ее последствия.

Индикаторы мультиколлинеарности:

1. Высокий R² и незначимые коэффициенты.
2. Сильная парная корреляция предикторов.
3. Сильные частные корреляции предикторов.
4. Высокий VIF – variance inflation factor.

Confidence interval (CI) is a type of [interval estimate](#) (of a [population parameter](#)) that is computed from the observed data. The confidence level is the frequency (i.e., the proportion) of possible confidence intervals that contain the true value of their corresponding parameter. In other words, if confidence intervals are constructed using a given confidence level in an infinite number of independent experiments, the proportion of those intervals that contain the true value of the parameter will match the confidence level. Confidence intervals consist of a range of values (interval) that act as good estimates of the unknown [population parameter](#). However, the interval computed from a particular sample does not necessarily include the true value of the parameter. Since the observed data are random samples from the true population, the confidence interval obtained from the data is also random. If a corresponding hypothesis test is performed, the confidence level is the complement of the level of [significance](#), i.e. a 95% confidence interval reflects a significance level of 0.05. If it is [hypothesized](#) that a true parameter value is 0 but the 95% confidence interval does not contain 0, then the estimate is [significantly different](#) from zero at the 5% significance level.

The desired level of confidence is set by the researcher (not determined by data). Most commonly, the 95% confidence level is used. However, other confidence levels can be used, for example, 90% and 99%. Factors affecting the width of the confidence interval include the size of the sample, the confidence level, and the variability in the sample. A larger sample size normally will lead to a better estimate of the population parameter.

A Confidence Interval is a range of values we are fairly sure our true value lies in.

$$\bar{X} \pm Z \cdot s / \sqrt{n}$$

X is the mean, Z is the chosen Z-value from the table, s is the standard deviation, n is the number of samples. The value after the ± is called the margin of error.

What's the difference between Gaussian Mixture Model and K-Means?

Let's say we are aiming to break them into three clusters, as above. K means will start with the assumption that a given data point belongs to one cluster.

Choose a data point. At a given point in the algorithm, we are certain that a point belongs to a red cluster. In the next iteration, we might revise that belief, and be certain that it belongs to the green cluster. However, remember, in each iteration, we are absolutely certain as to which cluster the point belongs to. This is the "hard assignment".

What if we are uncertain? What if we think, well, I can't be sure, but there is 70% chance it belongs to the red cluster, but also 10% chance it's in green, 20% chance it might be blue. That's a soft assignment. The Mixture of Gaussian model helps us to express this uncertainty. It starts with some prior belief about how certain we are about each point's cluster assignments. As it goes on, it revises those beliefs. But it incorporates the degree of uncertainty we have about our assignment.

Kmeans: find k to minimize $(x - \mu_k)^2$

Gaussian Mixture (EM clustering) : find k to minimize $(x - \mu_k)^2 / \sigma^2$

The difference (mathematically) is the denominator " σ^2 ", which means GM takes variance into consideration when it calculates the measurement.

Kmeans only calculates conventional Euclidean distance.

In other words, Kmeans calculate distance, while GM calculates "weighted" distance.

K means

1. Hard assign a data point to one particular cluster on convergence.
2. It makes use of the L2 norm when optimizing (Min {Theta} L2 norm point and its centroid coordinates).

EM

1. Soft assigns a point to clusters (so it give a probability of any point belonging to any centroid).
2. It doesn't depend on the L2 norm, but is based on the Expectation, i.e., the probability of the point belonging to a particular cluster. This makes K-means biased towards spherical clusters.

Describe how Gradient Boosting works.

The idea of boosting came out of the idea of whether a weak learner can be modified to become better.

Gradient boosting relies on regression trees (even when solving a classification problem) which minimize *mean squared error* (MSE). Selecting a prediction for a leaf region is simple: to minimize MSE we should select an average target value over samples in the leaf. The tree is built greedily starting from the root: for each leaf a split is selected to minimize MSE for this step.

To begin with, gradient boosting is an ensembling technique, which means that prediction is done by an ensemble of simpler estimators. While this theoretical framework makes it possible to create an ensemble of various estimators, in practice we almost always use GBDT — gradient boosting over decision trees.

The aim of gradient boosting is to create (or "train") an ensemble of trees, given that we know how to train a single decision tree. This technique is called **boosting** because we expect an ensemble to work much better than a single estimator.

Here comes the most interesting part. Gradient boosting builds an ensemble of trees **one-by-one**, then the predictions of the individual trees **are summed**: $D(x) = d_{\text{tree } 1}(x) + d_{\text{tree } 2}(x) + \dots$

The next decision tree tries to cover the discrepancy between the target function $f(x)$ and the current ensemble prediction **by reconstructing the residual**.

For example, if an ensemble has 3 trees the prediction of that ensemble is:

$D(x) = d_{\text{tree } 1}(x) + d_{\text{tree } 2}(x) + d_{\text{tree } 3}(x)$. The next tree (tree 4) in the ensemble should complement well the existing trees and minimize the training error of the ensemble.

In the ideal case we'd be happy to have: $D(x) + d_{\text{tree } 4}(x) = f(x)$.

To get a bit closer to the destination, we train a tree to reconstruct the difference between the target function and the current predictions of an ensemble, which is called the **residual**: $R(x) = f(x) - D(x)$. Did you notice? If decision tree completely reconstructs $R(x)$, the whole ensemble gives predictions without errors (after adding the newly-trained tree to the ensemble)! That said, in practice this never happens, so we instead continue the iterative process of ensemble building.

AdaBoost the First Boosting Algorithm

The weak learners in AdaBoost are decision trees with a single split, called decision stumps for their shortness.

AdaBoost works by weighting the observations, putting more weight on difficult to classify instances and less on those already handled well. New weak learners are added sequentially that focus their training on the more difficult patterns.

Gradient boosting involves three elements:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

1. Loss Function

The loss function used depends on the type of problem being solved.

It must be differentiable, but many standard loss functions are supported and you can define your own.

For example, regression may use a squared error and classification may use logarithmic loss.

A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

2. Weak Learner

Decision trees are used as the weak learner in gradient boosting.

Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and "correct" the residuals in the predictions.

Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss.

Initially, such as in the case of AdaBoost, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels.

It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes.

This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

3. Additive Model

Trees are added one at a time, and existing trees in the model are not changed.

A gradient descent procedure is used to minimize the loss when adding trees.

Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.

Instead of parameters, we have weak learner sub-models or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must add a tree to the model that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by reducing the residual loss.

Generally this approach is called functional gradient descent or gradient descent with functions.

The output for the new tree is then added to the output of the existing sequence of trees in an effort to correct or improve the final output of the model.

A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

Improvements to Basic Gradient Boosting

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly.

It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting.

In this section we will look at 4 enhancements to basic gradient boosting:

1. Tree Constraints
2. Shrinkage
3. Random sampling
4. Penalized Learning

1. Tree Constraints

It is important that the weak learners have skill but remain weak.

There are a number of ways that the trees can be constrained.

A good general heuristic is that the more constrained tree creation is, the more trees you will need in the model, and the reverse, where less constrained individual trees, the fewer trees that will be required.

Below are some constraints that can be imposed on the construction of decision trees:

- Number of trees, generally adding more trees to the model can be very slow to overfit. The advice is to keep adding trees until no further improvement is observed.
- Tree depth, deeper trees are more complex trees and shorter trees are preferred. Generally, better results are seen with 4-8 levels.
- Number of nodes or number of leaves, like depth, this can constrain the size of the tree, but is not constrained to a symmetrical structure if other constraints are used.

- Number of observations per split imposes a minimum constraint on the amount of training data at a training node before a split can be considered
- Minimum improvement to loss is a constraint on the improvement of any split added to a tree.

2. Weighted Updates

The predictions of each tree are added together sequentially.

The contribution of each tree to this sum can be weighted to slow down the learning by the algorithm. This weighting is called a shrinkage or a learning rate.

Each update is simply scaled by the value of the “learning rate parameter v ”

The effect is that learning is slowed down, in turn require more trees to be added to the model, in turn taking longer to train, providing a configuration trade-off between the number of trees and learning rate.

Decreasing the value of v [the learning rate] increases the best value for M [the number of trees].

It is common to have small values in the range of 0.1 to 0.3, as well as values less than 0.1.

Similar to a learning rate in stochastic optimization, shrinkage reduces the influence of each individual tree and leaves space for future trees to improve the model.

3. Stochastic Gradient Boosting

A big insight into bagging ensembles and random forest was allowing trees to be greedily created from subsamples of the training dataset.

This same benefit can be used to reduce the correlation between the trees in the sequence in gradient boosting models.

This variation of boosting is called stochastic gradient boosting.

at each iteration a subsample of the training data is drawn at random (without replacement) from the full training dataset. The randomly selected subsample is then used, instead of the full sample, to fit the base learner.

A few variants of stochastic boosting that can be used:

- Subsample rows before creating each tree.
- Subsample columns before creating each tree
- Subsample columns before considering each split.

Generally, aggressive sub-sampling such as selecting only 50% of the data has shown to be beneficial.

According to user feedback, using column sub-sampling prevents over-fitting even more so than the traditional row sub-sampling.

4. Penalized Gradient Boosting

Additional constraints can be imposed on the parameterized trees in addition to their structure.

Classical decision trees like CART are not used as weak learners, instead a modified form called a regression tree is used that has numeric values in the leaf nodes (also called terminal nodes). The values in the leaves of the trees can be called weights in some literature.

As such, the leaf weight values of the trees can be regularized using popular regularization functions, such as:

- L1 regularization of weights.
- L2 regularization of weights.

The additional regularization term helps to smooth the final learnt weights to avoid over-fitting. Intuitively, the regularized objective will tend to select a model employing simple and predictive functions.

В чем разница между AdaBoost & XGBoost?

<http://telegra.ph/Raznica-mezhdu-adaboost-i-XGBoost-10-18>

Data Mining Describe the decision tree model.

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

Each internal node represents a test on an attribute. Each leaf node represents a class.

The benefits of having a decision tree are as follows –

It does not require any domain knowledge.

It is easy to comprehend.

The learning and classification steps of a decision tree are simple and fast.

Tree Pruning

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

Tree Pruning Approaches

Here is the Tree Pruning Approaches listed below –

Pre-pruning – The tree is pruned by halting its construction early.

Post-pruning - This approach removes a sub-tree from a fully grown tree.

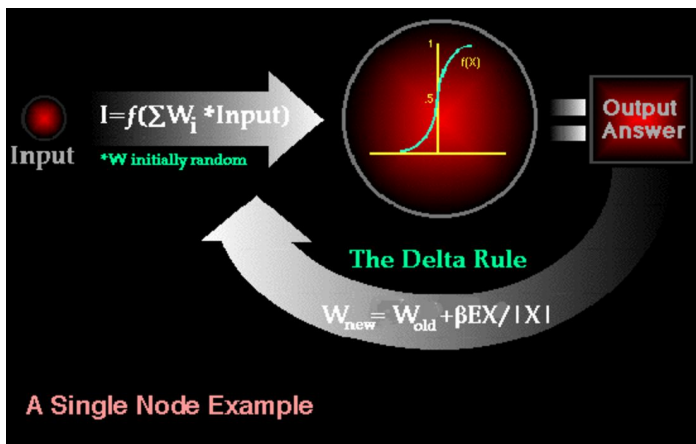
Cost Complexity

The cost complexity is measured by the following two parameters – Number of leaves in the tree, and Error rate of the tree.

Data Mining What is a neural network?

Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output as shown in the graphic below.

Although there are many different kinds of learning rules used by neural networks, this demonstration is concerned only with one: the delta rule. The delta rule is often utilized by the most common class of ANNs called 'backpropagation neural networks' (BPNNs). Backpropagation is an abbreviation for the backwards propagation of error. With the delta rule, as with other types of back propagation, 'learning' is a supervised process that occurs with each cycle or 'epoch' (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments. More simply, when a neural network is initially presented with a pattern it makes a random 'guess' as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights. More graphically, the process looks something like this:



Backpropagation performs a gradient descent within the solution's vector space towards a 'global minimum' along the steepest vector of the error surface. The global minimum is that theoretical solution with the lowest possible error. The error surface itself is a hyperparaboloid but is seldom 'smooth' as is depicted in the graphic below. Indeed, in most problems, the solution space is quite irregular with numerous 'pits' and 'hills' which may cause the network to settle down in a 'local minimum' which is not the best overall solution.

Since the nature of the error space can not be known a priori, neural network analysis often requires a large number of individual runs to determine the best solution. Most learning rules have built-in mathematical terms to assist in this process which control the 'speed' (Beta-coefficient) and the 'momentum' of the learning. The speed of learning is actually the rate of convergence between the current solution and the global minimum. Momentum helps the network to overcome obstacles (local minima) in the error surface and settle down at or near the global minimum.

Once a neural network is 'trained' to a satisfactory level it may be used as an analytical tool on other data. To do this, the user no longer specifies any training runs and instead allows the network to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is retained and no backpropagation occurs. The output of a forward propagation run is the predicted model for the data which can then be used for further analysis and interpretation.

How do you deal with sparse data?

We could take a look at L1 regularization since it best fits to the sparse data and do feature selection. If linear relationship - linear regression either - svm. Also it would be nice to use one-hot-encoding or bag-of-words.

A one hot encoding is a representation of categorical variables as binary vectors.

This first requires that the categorical values be mapped to integer values.

Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

Name and describe three different kernel functions and in what situation you would use each.

1. The Linear kernel is the simplest kernel function. It is given by the inner product $\langle x, y \rangle$ plus an

optional constant c . $k(x, y) = x^T y + c$ linear model. The boundary by using linear kernel is a linear.

2. The Polynomial kernel is a non-stationary kernel. Polynomial kernels are well suited for problems

where all the training data is normalized. $k(x, y) = (\alpha x^T y + c)^d$

In [machine learning](#), the polynomial kernel is a [kernel function](#) commonly used with [support vector machines](#)(SVMs) and other [kernelized](#) models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models. Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these. In the context of [regression analysis](#), such combinations are known as interaction features. The (implicit) feature space of a polynomial kernel is equivalent to that of [polynomial regression](#), but without the combinatorial blowup in the number of parameters to be learned. When the input features are binary-valued (booleans), then the features correspond to [logical conjunctions](#) of input features. It can be curve based on the degree of the polynomial.

3. The Gaussian kernel is an example of radial basis function kernel (rbf). $k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$

Alternatively, it could also be implemented using $k(x, y) = \exp(-\gamma\|x - y\|^2)$. The adjustable parameter sigma plays a major role in the performance of the kernel, and should be carefully tuned to the problem at hand. If overestimated, the exponential will behave almost linearly and the higher-dimensional projection will start to lose its non-linear power. In the other hand, if underestimated, the function will lack regularization and the decision boundary will be highly sensitive to noise in training data. In particular, it is commonly used in [support vector machine classification](#).

4. The exponential kernel is closely related to the Gaussian kernel, with only the square of the norm

left out. It is also a radial basis function kernel. $k(x, y) = \exp\left(-\frac{\|x - y\|}{2\sigma^2}\right)$

5. The Hyperbolic Tangent Kernel is also known as the Sigmoid Kernel and as the Multilayer Perceptron (MLP) kernel. The Sigmoid Kernel comes from the [Neural Networks](#) field, where the bipolar sigmoid function is often used as an [activation function](#) for artificial neurons.

$k(x, y) = \tanh(\alpha x^T y + c)$ It is interesting to note that a SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network. This kernel was quite popular for support vector machines due to its origin from neural network theory. Also, despite being only conditionally positive definite, it has been found to [perform well in practice](#). There are two adjustable parameters in the sigmoid kernel, the slope alpha and the intercept constant c. A common value for alpha is 1/N, where N is the data dimension. It is similar to the logistic regression model where a logistic function is used to define curves according to where the logistic value is greater than some value (modeling probability), such as 0.5 like the normal case.

Explain overfitting and how do you prevent overfitting?

1. collect more data
2. use ensembling methods that “average” models
3. choose simpler models
4. penalize complexity

For the first point, it may help to plot learning curves, plotting the training vs. the validation or cross-validation performance. If you see a trend that more data helps with closing the gap between the two, and if you could afford collecting more data, then this would probably be the best choice.

In my experience, ensembling is probably the most convenient way to build robust predictive models on somewhat small-sized datasets. As in real life, consulting a bunch of “experts” is usually not a bad idea before making a decision ;).

Regarding the third point, I usually start a predictive modeling task with the simplest model as a benchmark: usually logistic regression. Overfitting can be a real problem if our model has too much capacity — too many model parameters to fit, and too many hyperparameters to tune. If the dataset is small, a simple model is always a good option to prevent overfitting, and it is also a good benchmark for comparison to more “complex” alternatives.

Cross-Validation

A standard way to find out-of-sample prediction error is to use 5-fold cross validation. In this case, you can try 4-fold cross validation. Use 18 data points to build the model, and then evaluate the error on the 6 data points left out of the model. Repeat this 4 times, using a different set of 6 data points each time. This is better representative of the error you would expect when you're trying to predict a future value, rather than just how well you can fit the data at hand.

How do you deal with outliers in your data?

For the most part, if your data is affected by these extreme cases, you can bound the input to a historical representative of your data that excludes outliers. So that could be a number of items (>3) or a lower or upper bounds on your order value.

If the outliers are from a data set that is relatively unique then analyze them for your specific situation. Analyze both with and without them, and perhaps with a replacement alternative, if you have a reason for one, and report your results of this assessment.

One option is to try a transformation. Square root and log transformations both pull in high numbers. This can make assumptions work better if the outlier is a dependent.

What's the difference between supervised learning and unsupervised learning?

Supervised: So, if you are training your machine learning task for every input with corresponding target, it is called [supervised learning](#), which will be able to provide target for any new input after sufficient training. Your learning algorithm seeks a function from inputs to the respective targets. If the targets are expressed

in some classes, it is called classification problem. Alternatively, if the target space is continuous, it is called regression problem.

Unsupervised: Contrary, if you are training your machine learning task only with a set of inputs, it is called unsupervised learning, which will be able to find the structure or relationships between different inputs. Most important unsupervised learning is clustering, which will create different cluster of inputs and will be able to put any new input in appropriate cluster. Other than clustering, other unsupervised learning techniques are: [anomaly detection](#), [Hebbian Learning](#) and learning [Latent variable models](#) such as [Expectation–maximization algorithm](#), [Method of moments](#) ([mean](#), [covariance](#)) and [Blind signal separation](#) techniques ([Principal component analysis](#), [Independent component analysis](#), [Non-negative matrix factorization](#), [Singular value decomposition](#)) .

What is cross-validation and why would you use it?

Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

In k-fold cross-validation, the original sample is partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

For classification problems, one typically uses stratified k-fold cross-validation, in which the folds are selected so that each fold contains roughly the same proportions of class labels.

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data (as it is the case when fixing an arbitrary test set), which is a major advantage in problem such as inverse inference where the number of samples is very small.

What's the name of the matrix used to evaluate predictive models?

Confusion matrix.

What's the relationship between Principal Component Analysis (PCA) and Linear & Quadratic Discriminant Analysis (LDA & QDA)

[discriminant_analysis.LinearDiscriminantAnalysis](#) can be used to perform supervised dimensionality reduction, by projecting the input data to a linear subspace consisting of the directions which maximize the separation between classes (in a precise sense discussed in the mathematics section below). The dimension of the output is necessarily less than the number of classes, so this is in general a rather strong dimensionality reduction, and only makes sense in a multiclass setting.

In LDA we assume those Gaussian distributions for different classes share the same covariance structure. In **Quadratic Discriminant Analysis** (QDA) we don't have such a constraint. In LDA, as we mentioned, you simply assume for different k that the covariance matrix is identical. By making this assumption, the classifier becomes linear. The only difference from quadratic discriminant analysis is that we do not assume that the covariance matrix is identical for different classes. For QDA, the decision boundary is determined by a quadratic function.

QDA is not really that much different from LDA except that you assume that the covariance matrix can be different for each class and so, we will estimate the covariance matrix Σ_k separately for each class k , $k = 1, 2, \dots, K$.

This quadratic discriminant function is very much like the linear discriminant function except that because Σ_k , the covariance matrix, is not identical, you cannot throw away the quadratic terms. This discriminant function is a quadratic function and will contain second order terms.

LDA is also closely related to principal component analysis (PCA) and factor analysis in that both look for linear combinations of variables which best explain the data. LDA explicitly attempts to model the difference between the classes of data. PCA on the other hand does not take into account any difference in class.

PCA vs SVD

<https://www.quora.com/What-is-an-intuitive-explanation-of-the-relation-between-PCA-and-SVD/answer/Mike-Tamir>

If you had a categorical dependent variable and a mixture of categorical and continuous independent variables, what algorithms, methods, or tools would you use for analysis?

Logistic regression. If the dependent variable has 2 categories use a Binary Logistic Model. If the dependent variable has more than 2 categories use a Multinomial Logistic Model.

However, do note that **most software requires you to recode categorical predictors to a binary numeric system**. This just means coding sex to 0 for females and 1 for males or vice versa. For categorical variables with more than 2 levels you'll need to recode these into $L-1$ dummy variables where L is the number of levels and these dummies contain a 0 or 1 when they are in the corresponding category. This way each individual (sample) should be represented by having a 1 for the dummy variable he/she is part of and a 0 for the others, or a 0 for all dummies when he/she is part of the reference group.

What's the difference between logistic and linear regression? How do you avoid local minima?

The stochastic gradient (SG) algorithm behaves like a simulated annealing (SA) algorithm, where the learning rate of the SG is related to the temperature of SA. The randomness or noise introduced by SG

allows to escape from local minima to reach a better minimum. Of course, it depends on how fast you decrease the learning rate.

The path of stochastic gradient descent wanders over more places, and thus is more likely to "jump out" of a local minimum, and find a global minimum (Note*). However, stochastic gradient descent can still get stuck in local minimum.

Note: It is common to keep the learning rate constant, in this case stochastic gradient descent does not converge; it just wanders around the same point. However, if the learning rate decreases over time, say, it is inversely related to number of iterations then stochastic gradient descent would converge.

The parameters are estimated for every observation, this gives SGD more randomness, and therefore it is more likely to jump out of a local minimum.

Согласно принципу минимизации эмпирического риска для этого достаточно решить

оптимизационную задачу:
$$Q(w) = \sum_{i=1}^l L(a(x_i, w), y_i) \rightarrow \min_w$$
, где $L(a, y)$ - заданная функция потерь.

Для минимизации применим метод *градиентного спуска (gradient descent)*. Это пошаговый алгоритм, на каждой итерации которого вектор w изменяется в направлении наибольшего убывания функционала Q (то есть в направлении антиградиента):

$$w := w - \eta \nabla Q(w),$$

где η - положительный параметр, называемый *темпом обучения (learning rate)*.

Возможно 2 основных подхода к реализации градиентного спуска:

- *Пакетный (batch)*, когда на каждой итерации обучающая выборка просматривается целиком, и только после этого изменяется w . Это требует больших вычислительных затрат.
- *Стохастический (stochastic/online)*, когда на каждой итерации алгоритма из обучающей выборки каким-то (случайным) образом выбирается только один объект. Таким образом вектор w настраивается на каждый вновь выбираемый объект.

Explain a machine learning algorithm as if you're talking to a non-technical person.

Now you perform CLUSTERING.

You will find out the transactions/the people/ the products which are related to a group. For example, the people who buy stationary from your store will be in one cluster. You can use this information to see cool information like People who buy expensive pads buy cheap pens, and don't buy anything else which a regular housewife buys.

Then comes CLASSIFICATION

If a person who is 23 years old, doesn't have a job and doesn't earn much comes to your store to look for a new computer, he's not going to buy it.

ASSOCIATION MINING

A person who brought an expensive \$100 pen mostly purchases a few novels too. You can use this information to modify your store so that the pens section is far apart from novels section. The person will have to travel a long way over multiple aisles which may cause them to buy another thing on the way.

REGRESSION

When you had put up a Christmas sale with 20% discount on all products for kids, you had an surplus profit of \$50,000 in the last year. Depending on the products and their quantities/costs available right now, this year you can earn up to \$60,000 of profit even with a discount offer of 30%.

How would you build a model to predict credit card fraud?

Techniques used for fraud detection fall into two primary classes: statistical techniques and [artificial intelligence](#). Examples of statistical data analysis techniques are:

- [Data preprocessing](#) techniques for detection, validation, [error correction](#), and filling up of missing or incorrect data.
- Calculation of various statistical parameters such as [averages](#), [quantiles](#), performance metrics, probability distributions, and so on. For example, the averages may include average length of call, average number of calls per month and average delays in bill payment.
- Models and probability distributions of various business activities either in terms of various parameters or probability distributions.
- Computing [user profiles](#).
- Time-series analysis of time-dependent data.
- [Clustering](#) and [classification](#) to find patterns and [associations](#) among groups of data.
- [Matching algorithms](#) to [detect anomalies](#) in the behavior of transactions or users as compared to previously known models and profiles. Techniques are also needed to eliminate [false alarms](#), estimate risks, and predict future of current transactions or users.

Fraud management is a knowledge-intensive activity. The main AI techniques used for fraud management include:

- [Data mining](#) to classify, cluster, and [segment](#) the data and automatically find associations and rules in the data that may signify interesting patterns, including those related to fraud.
- [Expert systems](#) to encode expertise for detecting fraud in the form of rules.
- [Pattern recognition](#) to detect approximate classes, clusters, or patterns of suspicious behavior either automatically (unsupervised) or to match given inputs.
- Machine learning techniques to automatically identify characteristics of fraud.
- Neural networks that can learn suspicious patterns from samples and used later to detect them.

Solid feature engineering is extremely important for spotting credit card fraudsters. The relative imbalance of fraud in a dataset can start at 1 fraudster to 1000 good people. So it's an extremely highly imbalanced classification problem. The extent to which you can build good features is predicated on having good ground truth data, and a rich data set to deal with. You can catch a large percentage of fraudsters with a

small amount of relatively simple features. You'll still have a high false positive rate at this point as your model won't be rich enough to capture nuances. Furthermore, fraudsters change tactics when they notice that old tactics aren't working anymore: you need to make sure that you are encapsulating the behaviour behind new attack vectors constantly.

Naturally, the actual models that you build to detect fraud are statistically based. From my personal experience, the popular models to choose tend to be logistic regression, Bayesian models and Random Forests. I bias towards explainable models: in fraud detection and money laundering, it's usually mandatory to tell an analyst **why** you decided to block a particular card or customer. The ability to give detailed feedback as to why the analyst thinks you are wrong can also help you improve your model quickly and fight false positives. It's incredibly beneficial, both from a user experience perspective, and the ability to understand the reasons for false positives.

There are a few other statistical and non-statistical approaches that are useful. Unsupervised approaches for anomaly detection can prove extremely useful: fraudsters are hard to find, but tend to look a bit weird compared to the general population. Deep learning, although not really statistically based, can also be useful if you have enough labeled data (i.e. PayPal, Google, big banks). As a business, it's hard to get enough labels here unless your business is wildly successful or you're agglomerating data across clients like we are. PCA and TSNE are also useful in feature engineering, allowing you to view clusters of high dimensional customer behaviour to gain an understanding of the distribution of behaviour among different groups of customers.

But from an algorithmic perspective, what makes credit card fraud detection unique is that you have so few "negative" labels. Fraud makes up <1% of transactions, and every fraudster that actually makes it through is costly. So while you can use some supervised classification techniques like logistic regression, you need to pair those with anomaly detection algorithms that can identify outliers without much training data.

For example, you might try training a one-class SVM, which will assign a probability to each transaction. Transactions with very low probabilities might indicate suspicious behavior, which should be reviewed and approved/denied. That can operate without any labelled data.

Other approaches include k-nearest neighbors, clustering, or semi-supervised learning.

How do you handle missing or bad data?

First, determine the pattern of your missing data. There are three types of missing data:

- Missing Completely at Random: There is no pattern in the missing data on any variables. This is the best you can hope for.
- Missing at Random: There is a pattern in the missing data but not on your primary dependent variables such as likelihood to recommend or SUS Scores.
- Missing Not at Random: There is a pattern in the missing data that affects your primary dependent variables. For example, lower-income participants are less likely to respond and

thus affect your conclusions about income and likelihood to recommend. Missing not at random is your worst-case scenario. Proceed with caution.

And here are seven things you can do about that missing data:

1. Listwise Deletion: Delete all data from any participant with missing values. If your sample is large enough, then you likely can drop data without substantial loss of statistical power. Be sure that the values are missing at random and that you are not inadvertently removing a class of participants.¹
2. Recover the Values: You can sometimes contact the participants and ask them to fill out the missing values. For in-person studies, we've found having an additional check for missing values before the participant leaves helps. Mean, median

If you're attempting to predict a customer's gender, and you only have 100 data points, what problems could arise?

- Over-fitting becomes much harder to avoid
- You don't only over-fit to your training data, but sometimes you over-fit to your validation set as well.
- Outliers become much more dangerous.
- Noise in general becomes a real issue, be it in your target variable or in some of the features.

Describe a non-normal probability distribution and how to apply it

1. Beta Distribution.
2. Exponential Distribution.
3. Gamma Distribution.
4. Inverse Gamma Distribution.
5. Log Normal Distribution.
6. Logistic Distribution.
7. Maxwell-Boltzmann Distribution.
8. Poisson Distribution.
9. Skewed Distribution.
10. Symmetric Distribution.
11. Uniform Distribution.
12. Unimodal Distribution.
13. Weibull Distribution.

You have several options for handling your non normal data. Several tests, including the one sample Z test, T test and ANOVA assume normality. You may still be able to run these tests if your sample size is large enough (usually over 20 items). You can also choose to transform the data with a function, forcing it to fit a normal model. However, if you have a very small sample, a sample that is skewed or one that naturally fits another distribution type, you may want to run a non parametric test. A non parametric test is

one that doesn't assume the data fits a specific distribution type. Non parametric tests include the Wilcoxon signed rank test, the Mann-Whitney U Test and the Kruskal-Wallis test.

Extreme Values

Too many extreme values in a data set will result in a skewed distribution. Normality of data can be achieved by cleaning the data. This involves determining measurement errors, data-entry errors and outliers, and removing them from the data for valid reasons.

It is important that outliers are identified as truly special causes before they are eliminated. Never forget: The nature of normally distributed data is that a small percentage of extreme values can be expected; not every outlier is caused by a special reason. Extreme values should only be explained and removed from the data if there are more of them than expected under normal conditions.

Overlap of Two or More Processes

Data may not be normally distributed because it actually comes from more than one process, operator or shift, or from a process that frequently shifts. If two or more data sets that would be normally distributed on their own are overlapped, data may look bimodal or multimodal – it will have two or more most-frequent values.

The remedial action for these situations is to determine which X's cause bimodal or multimodal distribution and then **stratify** the data. The data should be checked again for normality and afterward the stratified processes can be worked with separately.

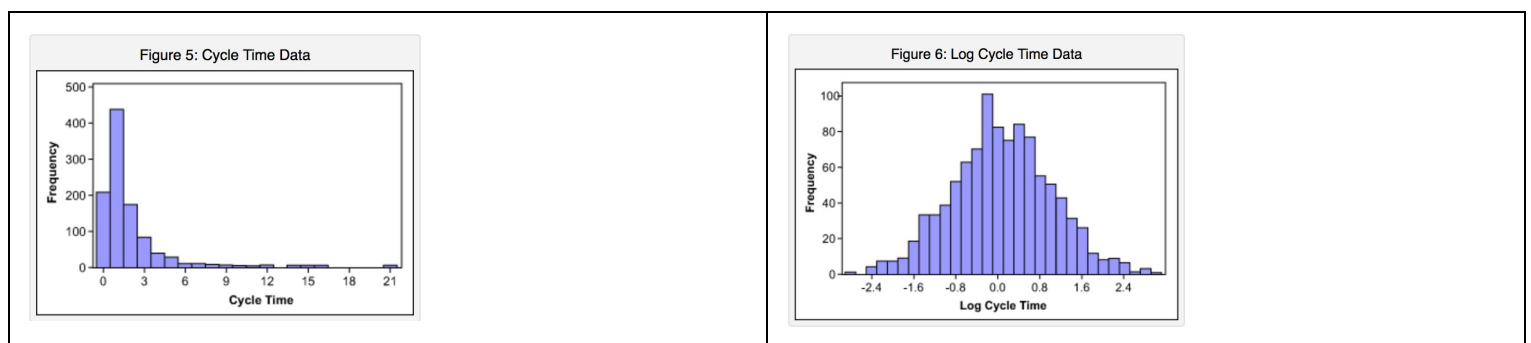
Insufficient Data Discrimination

Round-off errors or measurement devices with poor resolution can make truly continuous and normally distributed data look discrete and not normal. Insufficient data discrimination – and therefore an insufficient number of different values – can be overcome by using more accurate measurement systems or by collecting more data.

Values Close to Zero or a Natural Limit

If a process has many values close to zero or a natural limit, the data distribution will skew to the right or left. In this case, a transformation, such as the Box-Cox power transformation, may help make data normal. In this method, all data is raised, or transformed, to a certain exponent, indicated by a Lambda value. When comparing transformed data, everything under comparison must be transformed in the same way.

The figures below illustrate an example of this concept. Figure 5 shows a set of cycle-time data; Figure 6 shows the same data transformed with the natural logarithm.



Take note: None of the transformation methods provide a guarantee of a normal distribution. Always check with a probability plot to determine whether normal distribution can be assumed after transformation.

Some statistical tools do not require normally distributed data. To help practitioners understand when and how these tools can be used, the table below shows a comparison of tools that do not require normal distribution with their normal-distribution equivalents.

Comparison of Statistical Analysis Tools for Normally and Non-Normally Distributed Data		
Tools for Normally Distributed Data	Equivalent Tools for Non-Normally Distributed Data	Distribution Required
T-test	Mann-Whitney test; Mood's median test; Kruskal-Wallis test	Any
ANOVA	Mood's median test; Kruskal-Wallis test	Any
Paired t-test	One-sample sign test	Any
F-test; Bartlett's test	Levene's test	Any
Individuals control chart	Run Chart	Any
C_p/C_{pk} analysis	C_p/C_{pk} analysis	Weibull; log-normal; largest extreme value; Poisson; exponential; binomial

Data Mining Explain what heteroskedasticity is and how to solve it

A scatterplot of these variables will often create a cone-like shape, as the scatter (or variability) of the dependent variable (DV) widens or narrows as the value of the independent variable (IV) increases. The inverse of heteroscedasticity is homoscedasticity, which indicates that a DV's variability is equal across values of an IV. More specifically, it is assumed that the error (a.k.a residual) of a regression model is homoscedastic across all values of the predicted value of the DV. Put more simply, a test of homoscedasticity of error terms determines whether a regression model's ability to predict a DV is consistent across all values of that DV. If a regression model is consistently accurate when it predicts low values of the DV, but highly inconsistent in accuracy when it predicts high values, then the results of that regression should not be trusted.

A classic example of heteroscedasticity is that of income versus expenditure on meals. As one's income increases, the variability of food consumption will increase. A poorer person will spend a rather constant amount by always eating inexpensive food; a wealthier person may occasionally buy inexpensive food and at other times eat expensive meals. Those with higher incomes display a greater variability of food consumption.

Re-build the model with new predictors. Variable transformation such as Box-Cox transformation.

There are four common corrections for heteroscedasticity. They are:

- View [logarithmized](#) data. Non-logarithmized series that are growing exponentially often appear to have increasing variability, random volatility, or volatility clusters as the series rises over time. The variability in percentage terms may, however, be rather stable. The reason for this is that the likelihood function for exponentially growing data lacks a variance. Using regression, the maximum likelihood estimator is the [least squares estimator](#), a form of the sample mean, but the sampling distribution of the estimator is the [Cauchy distribution](#). The Cauchy distribution has no variance and so there is no fixed point for the sample variance to converge to causing it to

behave as a random number. Taking the logarithm of the data converts the likelihood function to the [hyperbolic secant distribution](#), which has a defined variance.

- Use a different specification for the model (different X variables, or perhaps non-linear transformations of the X variables).
- Apply a [weighted least squares](#) estimation method, in which OLS is applied to transformed or weighted values of X and Y . The weights vary over observations, usually depending on the changing error variances. In one variation the weights are directly related to the magnitude of the dependent variable, and this corresponds to least squares percentage regression.
- [Heteroscedasticity-consistent standard errors](#) (HCSE), while still biased, improve upon OLS estimates.^[5] HCSE is a consistent estimator of standard errors in regression models with heteroscedasticity. This method corrects for heteroscedasticity without altering the values of the coefficients. This method may be superior to regular OLS because if heteroscedasticity is present it corrects for it, however, if the data is homoscedastic, the standard errors are equivalent to conventional standard errors estimated by OLS. Several modifications of the White method of computing heteroscedasticity-consistent standard errors have been proposed as corrections with superior finite sample properties.

What are some different Time Series forecasting techniques?+

Stationarity is defined using very strict criterion. However, for practical purposes we can assume the series to be stationary if it has constant statistical properties over time, ie. the following: constant mean, constant variance, autocovariance that does not depend on time.

Dickey-Fuller Test: This is one of the statistical tests for checking stationarity. Here the null hypothesis is that the TS is non-stationary. The test results comprise of a **Test Statistic** and some **Critical Values** for different confidence levels. If the 'Test Statistic' is less than the 'Critical Value', we can reject the null hypothesis and say that the series is stationary.

One of the first tricks to reduce trend can be **transformation**. These can be taking a log, square root, cube root, etc.

ARIMA. I won't go into the technical details but you should understand these concepts in detail if you wish to apply them more effectively. ARIMA stands for **Auto-Regressive Integrated Moving Averages**. The ARIMA forecasting for a stationary time series is nothing but a linear (like a linear regression) equation. The predictors depend on the parameters (p,d,q) of the ARIMA model:

1. **Number of AR (Auto-Regressive) terms (p):** AR terms are just lags of dependent variable. For instance if p is 5, the predictors for $x(t)$ will be $x(t-1) \dots x(t-5)$.
2. **Number of MA (Moving Average) terms (q):** MA terms are lagged forecast errors in prediction equation. For instance if q is 5, the predictors for $x(t)$ will be $e(t-1) \dots e(t-5)$ where $e(i)$ is the difference between the moving average at i th instant and actual value.
3. **Number of Differences (d):** These are the number of nonseasonal differences, i.e. in this case we took the first order difference. So either we can pass that variable and put $d=0$ or pass the original variable and put $d=1$. Both will generate same results.

An importance concern here is how to determine the value of 'p' and 'q'. We use two plots to determine these numbers. Lets discuss them first.

1. **Autocorrelation Function (ACF):** It is a measure of the correlation between the the TS with a lagged version of itself. For instance at lag 5, ACF would compare series at time instant 't1'...'t2' with series at instant 't1-5'...'t2-5' (t1-5 and t2 being end points).
2. **Partial Autocorrelation Function (PACF):** This measures the correlation between the TS with a lagged version of itself but after eliminating the variations already explained by the intervening comparisons. Eg at lag 5, it will check the correlation but remove the effects already explained by lags 1 to 4.

ARMA models are commonly used in time series modeling. In ARMA model, AR stands for auto-regression and MA stands for moving average. If these words sound intimidating to you, worry not – I'll simplify these concepts in next few minutes for you!

Difference between AR and MA models

The primary difference between an AR and MA model is based on the correlation between time series objects at different time points. The correlation between $x(t)$ and $x(t-n)$ for $n > \text{order of MA}$ is always zero. This directly flows from the fact that covariance between $x(t)$ and $x(t-n)$ is zero for MA models (something which we refer from the example taken in the previous section). However, the correlation of $x(t)$ and $x(t-n)$ gradually declines with n becoming larger in the AR model. This difference gets exploited irrespective of having the AR model or MA model. The correlation plot can give us the order of MA model.

ACF is a plot of total correlation between different lag functions. For instance, in GDP problem, the GDP at time point t is $x(t)$. We are interested in the correlation of $x(t)$ with $x(t-1)$, $x(t-2)$ and so on. Now let's reflect on what we have learnt above.

In a moving average series of lag n , we will not get any correlation between $x(t)$ and $x(t - n - 1)$. Hence, the total correlation chart cuts off at n th lag. So it becomes simple to find the lag for a MA series. For an AR series this correlation will gradually go down without any cut off value. So what do we do if it is an AR series?

Here is the second trick. If we find out the partial correlation of each lag, it will cut off after the degree of AR series. For instance, if we have a $AR(1)$ series, if we exclude the effect of 1st lag ($x(t-1)$), our 2nd lag ($x(t-2)$) is independent of $x(t)$. Hence, the partial correlation function (PACF) will drop sharply after the 1st lag.

Explain Principle Component Analysis (PCA) and equations PCA uses.+

Principal component analysis (PCA) is a statistical procedure that uses an [orthogonal transformation](#) to convert a set of observations of possibly correlated variables into a set of values of [linearly uncorrelated](#) variables called **principal components**. This transformation is defined in such a way that the first principal component has the largest possible [variance](#) (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is [orthogonal](#) to the preceding components. The resulting vectors are an uncorrelated [orthogonal basis set](#). PCA is sensitive to the relative scaling of the original variables.

PCA is mostly used as a tool in [exploratory data analysis](#) and for making [predictive models](#). It's often used to visualize genetic distance and relatedness between populations. PCA can be done by [eigenvalue](#)

[decomposition](#) of a data [covariance](#) (or [correlation](#)) matrix or [singular value decomposition](#) of a [data matrix](#), usually after mean centering (and normalizing or using [Z-scores](#)) the data matrix for each attribute.^[4] The results of a PCA are usually discussed in terms of component scores, sometimes called factor scores (the transformed variable values corresponding to a particular data point), and loadings (the weight by which each standardized original variable should be multiplied to get the component score).

PCA can be thought of as fitting an n -dimensional [ellipsoid](#) to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a commensurately small amount of information.

To find the axes of the ellipsoid, we must first subtract the mean of each variable from the dataset to center the data around the origin. Then, we compute the [covariance matrix](#) of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix. Then, we must orthogonalize the set of eigenvectors, and normalize each to become unit vectors. Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data. The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

This procedure is sensitive to the scaling of the data, and there is no consensus as to how to best scale the data to obtain optimal results.

PCA is mathematically defined as an [orthogonal linear transformation](#) that transforms the data to a new [coordinate system](#) such that the greatest variance by some projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.

First component

In order to maximize variance, the first loading vector $\mathbf{w}_{(1)}$ thus has to satisfy

$$\mathbf{w}_{(1)} = \arg \max_{\|\mathbf{w}\|=1} \left\{ \sum_i (t_{1(i)})^2 \right\} = \arg \max_{\|\mathbf{w}\|=1} \left\{ \sum_i (\mathbf{x}_{(i)} \cdot \mathbf{w})^2 \right\}$$

Equivalently, writing this in matrix form gives

$$\mathbf{w}_{(1)} = \arg \max_{\|\mathbf{w}\|=1} \{ \|\mathbf{X}\mathbf{w}\|^2 \} = \arg \max_{\|\mathbf{w}\|=1} \{ \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \}$$

Since $\mathbf{w}_{(1)}$ has been defined to be a unit vector, it equivalently also satisfies

$$\mathbf{w}_{(1)} = \arg \max \left\{ \frac{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

The quantity to be maximised can be recognised as a [Rayleigh quotient](#). A standard result for a [symmetric matrix](#) such as $\mathbf{X}^T \mathbf{X}$ is that the quotient's maximum possible value is the largest [eigenvalue](#) of the matrix, which occurs when \mathbf{w} is the corresponding [eigenvector](#).

With $\mathbf{w}_{(1)}$ found, the first component of a data vector $\mathbf{x}_{(i)}$ can then be given as a score $t_{1(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}$ in the transformed co-ordinates, or as the corresponding vector in the original variables, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(1)}\} \mathbf{w}_{(1)}$.

Further components

The k th component can be found by subtracting the first $k - 1$ principal components from

$$\hat{\mathbf{X}}_k = \mathbf{X} - \sum_{s=1}^{k-1} \mathbf{X} \mathbf{w}_{(s)} \mathbf{w}_{(s)}^T$$

and then finding the loading vector which extracts the maximum variance from this new data matrix

$$\mathbf{w}_{(k)} = \arg \max_{\|\mathbf{w}\|=1} \left\{ \|\hat{\mathbf{X}}_k \mathbf{w}\|^2 \right\} = \arg \max \left\{ \frac{\mathbf{w}^T \hat{\mathbf{X}}_k^T \hat{\mathbf{X}}_k \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

The components of \mathbf{C}_x , denoted by c_{ij} , represent the covariances between the random variable components x_i and x_j . The component c_{ii} is the variance of the component x_i . The variance of a component indicates the spread of the component values around its mean value. If two components x_i and x_j of the data are uncorrelated, their covariance is zero ($c_{ij} = c_{ji} = 0$). The covariance matrix is, by definition, always symmetric.

This means that we project the original data vector on the coordinate axes having the dimension K and transforming the vector back by a linear combination of the basis vectors. This minimizes the mean-square error between the data and this representation with given number of eigenvectors.

If the data is concentrated in a linear subspace, this provides a way to compress data without losing much information and simplifying the representation. By picking the eigenvectors having the largest eigenvalues we lose as little information as possible in the mean-square sense. One can e.g. choose a fixed number of eigenvectors and their respective eigenvalues and get a consistent representation, or abstraction of the data.

How do you solve Multicollinearity?+

In regression, "multicollinearity" refers to predictors that are correlated with other predictors.

Multicollinearity occurs when your model includes multiple factors that are correlated not just to your response variable, but also to each other. In other words, it results when you have factors that are a bit redundant.

Multicollinearity increases the standard errors of the coefficients. Increased standard errors in turn means that coefficients for some independent variables may be found not to be significantly different from 0. In other words, by overinflating the standard errors, multicollinearity makes some variables statistically insignificant when they should be significant. Without multicollinearity (and thus, with lower standard errors), those coefficients might be significant.

Solutions:

- **Remove highly correlated predictors from the model.** If you have two or more factors with a high VIF, remove one from the model. Because they supply redundant information, removing one of the correlated factors usually doesn't drastically reduce the R-squared. Consider using stepwise regression, best subsets regression, or specialized knowledge of the data set to remove these variables. Select the model that has the highest R-squared value.

- **Use Partial Least Squares Regression (PLS) or Principal Components Analysis**, regression methods that cut the number of predictors to a smaller set of uncorrelated components.

What are p-values and confidence intervals?+

The P value, or calculated probability, is the probability of finding the observed, or more extreme, results when the **null hypothesis (H_0)** of a study question is true – the definition of ‘extreme’ depends on how the hypothesis is being tested. P is also described in terms of rejecting H_0 when it is actually true, however, it is not a direct probability of this state.

If your P value is less than the chosen significance level then you reject the null hypothesis i.e. accept that your sample gives reasonable evidence to support the alternative hypothesis. It does NOT imply a "meaningful" or "important" difference; that is for you to decide when considering the real-world relevance of your result.

The choice of significance level at which you reject H_0 is arbitrary. Conventionally the 5% (less than 1 in 20 chance of being wrong), 1% and 0.1% ($P < 0.05$, 0.01 and 0.001) levels have been used. These numbers can give a false sense of security.

Most authors refer to **statistically significant** as $P < 0.05$ and **statistically highly significant** as $P < 0.001$ (less than one in a thousand chance of being wrong).

Data Analyst If you have 70 red marbles, and the ratio of green to red marbles is 2 to 7, how many green marbles are there?+

Green / red = 2 / 7

$X/70 = 2/7 \quad x = 2 \cdot 70/7 = 20$

Given a die, would it be more likely to get a single 6 in six rolls, at least two 6s in twelve rolls, or at least one-hundred 6s in six-hundred rolls?

<http://www.kdnuggets.com/2017/03/top-firms-100-data-science-interview-questions.html>

<https://www.springboard.com/blog/data-science-interview-questions/>

<https://www.dezyre.com/article/100-data-science-interview-questions-and-answers-general-for-2017/184>

An SVM model is suffering with under fitting.

In the case of underfitting, we need to increase the complexity of a model. When we increase the value of C, it means that we are making decision boundary more complex.

What's the Central Limit Theorem, and how do you prove it? What are its applications?

Suppose that we are interested in estimating the average height among all people. Collecting data for every person in the world is impractical, bordering on impossible. While we can't obtain a height measurement from everyone in the [population](#), we can still [sample](#) some people. The question now becomes, what can we say about the average height of the entire population given a single sample.

The Central Limit Theorem addresses this question exactly. Formally, it states that if we sample from a population using a sufficiently large sample size, the mean of the samples (also known as the sample

population) will be normally distributed (assuming true random sampling). What’s especially important is that this will be true regardless of the distribution of the original population.

The mean of the sampling distribution will approximate the mean of the true population distribution. Additionally, the variance of the sampling distribution is a function of both the population variance and the sample size used. A larger sample size will produce a smaller sampling distribution variance. This makes intuitive sense, as we are considering more samples when using a larger sample size, and are more likely to get a representative sample of the population. So roughly speaking, if the sample size used is large enough, there is a good chance that it will estimate the population pretty well. Most sources state that for most applications $N = 30$ is sufficient.

These principles can help us to reason about samples from any population. Depending on the scenario and the information available, the way that it is applied may vary. For example, in some situations we might know the true population mean and variance, which would allow us to compute the variance of any sampling distribution. However, in other situations, such as the original problem we discussed of estimating average human height, we won’t know the true population mean and variance. Understanding the nuances of sampling distributions and the Central Limit Theorem is an essential first step toward talking many of these problems.

Функция распределения

Функция распределения в [теории вероятностей](#) — функция, характеризующая [распределение случайной величины](#) или случайного вектора; вероятность того, что случайная величина X примет значение, меньшее или равное x , где x — произвольное действительное число. При соблюдении известных условий (см. [ниже](#)) полностью определяет случайную величину.

Statistical Interactions

Basically, an interaction is when the effect of one factor (input variable) on the dependent variable (output variable) differs among levels of another factor.

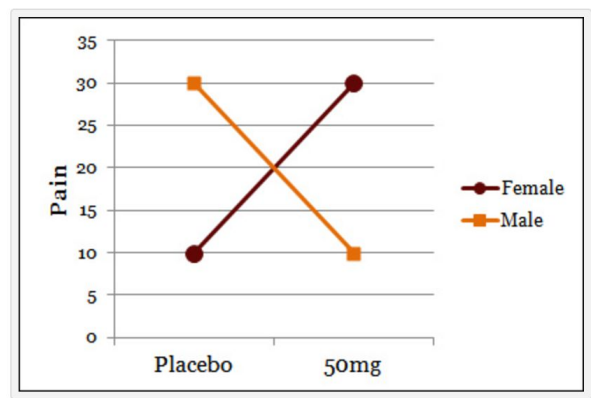
For example, in a pain relief drug trial, one factor is “dose” and another factor is “gender”. The dependent variable is “headache pain” (measured on a scale of 0 to 50). We can look at the findings by showing the means for each group in a table like this

	Male	Female
Placebo	30	10
50mg	10	30

If compare the “marginal means”, we can see that the average pain score for women was 20 and the average pain score for men was 20. There is no difference in headache pain between men and women. Likewise, the drug appears to have no effect on headache pain.

However, if we had stopped there, we would be missing some important findings. There is an interaction between gender and dose on headache pain. If we graph the means of each group, we can see it clearly: men have more headache pain than women unless they take the drug. There are simple effects of dose for both men and women, but the effects “wash out” one another. The result is no main effect of dose or

gender. If you did not examine gender, it would appear that your drug did not work; if you did not study drug dose, you would see no difference between men and women. Examining the interaction allows us to see that the drug works – for men – AND that it causes pain in women.

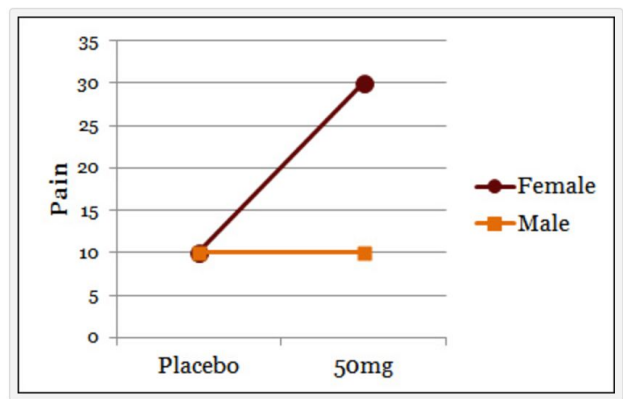


This is called a “pure interaction” because there are no main effects, but there is an interaction.

More commonly seen, however, when effects occur in both (or all) conditions, but they are stronger in one condition than another. Also fairly common is an effect in one condition (or for one group), but no effect at all in another. For example, what if our same pain study resulted in the following means:

	Male	Female
Placebo	10	10
50mg	10	30

For men, there is no effect of the drug on headache pain. The drug causes pain in women, who would otherwise have the same amount of pain as men.



To recap, an interaction is when the effect of one variable differs among levels of another variable. When interactions are seen among continuous variables (variables with a range of values, as opposed to categories), they look a little different, but the meaning is basically the same. In the last example, difference in pain between men and women (the male average was 10, the female average was 20) are driven by the interaction with the drug.

What error metric would you use to evaluate how good a binary classifier is?-

Assuming a clustering model's labels are known, how do you evaluate the performance of the model?-

Choose any machine learning algorithm and describe it.-

PCA, SVD; k means; decision tree;

Describe a method used in machine learning.-

How would you derive new features from features that already exist?-

Suppose you were given two years of transaction history. What features would you use to predict credit risk?-

Explain Cross-validation as if you're talking to a non-technical person-

Среднеквадратичное отклонение

Случайная величина

Закон больших чисел

Мат ожидание

Дисперсия

Корреляция, Коварияция

Градиентный спуск

Стохастический градиентный спуск

Локальный/глобальный минимум/максимум

Метод наименьших квадратов

Строго выпуклая функция

Несбалансированная выборка

Хи-квадрат

Дискретная величина

Непрерывная величина

Плотность вероятности

Функция плотности вероятности

Функция непрерывного распределения вероятности плотности.

Функция распределения случайной величины

Функция вероятности

Мультиколлинеарность

Распределения:

- Биномиальное
- Равномерное на отрезке
- Бернулли
- Нормальное (Гаусса)
- Пуассона

R2_score & cross_val_score должны стремиться к единице

критерий Шапиро-Уилка (Shapiro-Wilk)

критерий Колмогорова-Смирнова (Kolmogorov-Smirnov)

тренд, ARMA/ARIMA, спектральный анализ

Дискретное преобразование Фурье

Вейвлет-преобразование

Косинусное преобразование

Автокорреляция

Экспоненциальное сглаживание

Ортогональная матрица

difference between a histogram and a bar chart

Softmax

K-means

Back propagation

Xgboost

Реляционные и нереляционные БД. Разные join

<https://www.datacamp.com/community/tutorials/python-numpy-tutorial>

<https://www.datacamp.com/community/tutorials/machine-learning-python>

<https://www.springboard.com/blog/data-science-interview-questions/>

Линейная алгебра:

- Разложение матрицы на собственные значения матрицы, собственные вектора матрицы
- Базис матрицы
- Ортонормированный базис
- Диагональная матрица