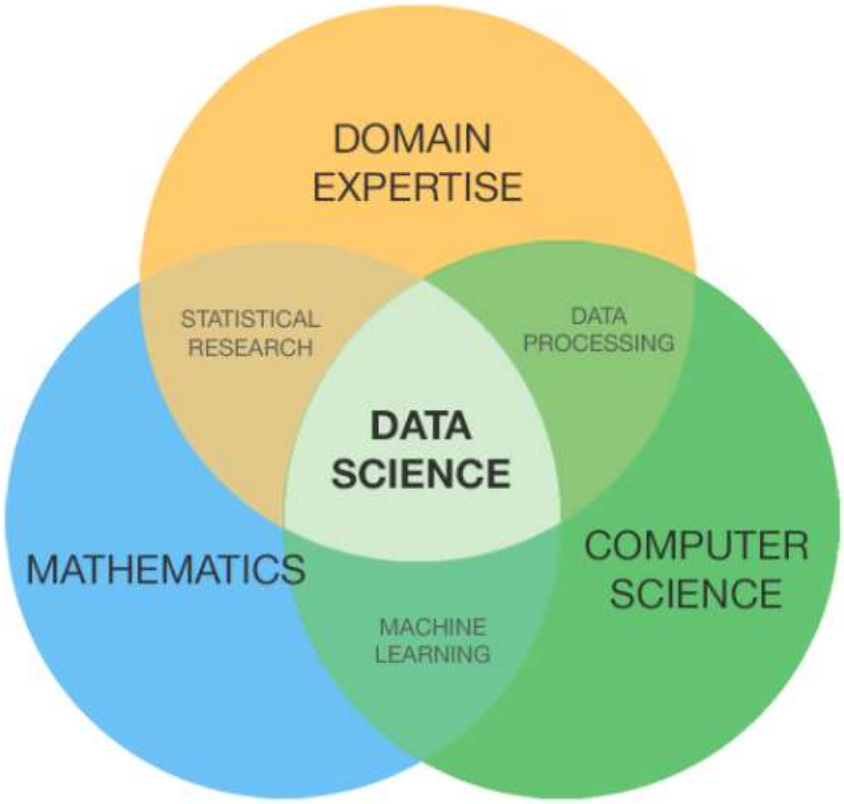
l blogs

by night.

Apr 3 · 12 min read

Data Science Interview Guide



Data Science is quite a large and diverse field. As a result, it is really difficult to be a jack of all trades. Traditionally, Data Science would focus on mathematics, computer science and domain expertise. While I will briefly cover some computer science fundamentals, the bulk of this blog will mostly cover the mathematical basics one might either need to brush up on (or even take an entire course).

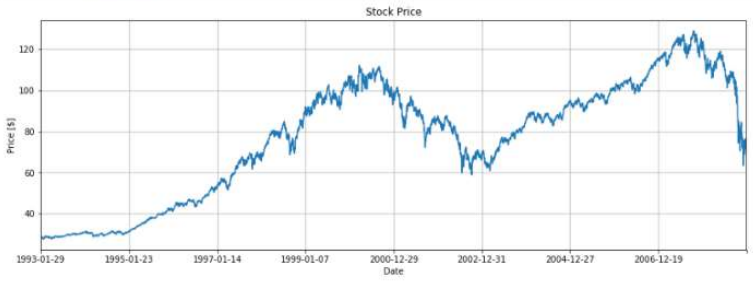
Software Tools

In most data science workplaces, software skills are a must. While I understand most of you reading this are more math heavy by nature, realize the bulk of data science (dare I say 80%+) is collecting, cleaning and processing data into a useful form.

Programming Language

Python and R are the most popular ones in the Data Science space. However, I have also come across C/C++, Java and Scala. Although, I would personally recommend Python as it has all the math libraries as well as specialized libraries for querying various databases and maintaining interactive web UIs. Common Python libraries of choice are matplotlib, numpy, pandas and scikit-learn.

```
In [4]: plt.figure(figsize=(15,5))
        df['Adj Close'].plot(grid=True)
        plt.ylabel('Price [$]')
        plt.title('Stock Price');
```



Database Management

It is common to see the majority of the data scientists being in one of two camps: Mathematicians and Database Architects. If you are the second one, the blog won't help you much (YOU ARE ALREADY AWESOME!). If you are among the first group (like me), chances are you feel that writing a double nested SQL query is an utter nightmare. That being said, it is important to have some knowledge of query optimization (both for SQL and noSQL systems).

```
In [5]: Bulbasaur = {'Name':'Bulbasaur',
                     'Type':'Grass'}

        Charmander = {'Name':'Charmander',
                      'Type':'Fire'}

        Squirtle = {'Name':'Squirtle',
                    'Type':'Water'}

        Pikachu = {'Name':'Pikachu',
                   'Type':'Electrical'}
```

```
In [6]: #Enter single document
        Pokemon_Collection.insert_one(Bulbasaur)

        #Enter Multiple documents
        Pokemon_Collection.insert_many(Charmander, Squirtle, Pikachu)
```

Map Reduce

Big Data technologies are a little hard to follow considering how the Apache project keeps on adding new tools all the time. However, I would recommend learning either Hadoop or Spark (though my personal recommendation is Spark). Both use similar Map Reduce algorithms (except Hadoop does it on disk while Spark does it in memory). Common Spark wrappers are Scala, Python and Java.

```
In [4]: for n in range(1,6):
            # Put your logic for generating the sorted n-gram RDD here and store it in freq_ngramRDD variable

            freq_ngramRDD = sentences.map(lambda x: remove_punc(x)) \       #remove punctuations
                            .flatMap(lambda x: ngrams(x.split(), n)) \     #arrange into n-grams
                            .map(lambda x: (x, 1)) \                       #format: (word, 1)
                            .reduceByKey(lambda x,y:x+y) \                 #add 1s by words to get count
                            .map(lambda x:(x[1],x[0])) \                   #format: (word, count)
                            .sortByKey(False)                              #sort descending order
            printOutput(n,freq_ngramRDD)                                   #print output
```

Additional Information

For more information on software development for data science applications, here are some of my other blogs:
- Python based Plotting with Matplotlib
- Python Package Management with Conda
- How to make your Software Development experience… painless….
- Software Development Design Principles

Data Collection And Cleaning

Now that we have covered the software needs, we will start making a smooth transition into the mathematics domain. Around this part of the process, you generally need to have some parsing background. This might either be collecting sensor data, parsing websites or carrying out surveys. After collecting the data, it needs to be transformed into a usable form (e.g. key-value store in JSON Lines files). Once the data is collected and put in a usable format, it is essential to perform some data quality checks. Some common quality checks are as described below:

NaN Handling

NaN or “Not A Number” is a common place holder for missing data. If the number of NaNs for the specific feature is small, it usually suffice fill in the NaNs with the average value (of the entire dataset or a window), or with 0s (for a sparse dataset).

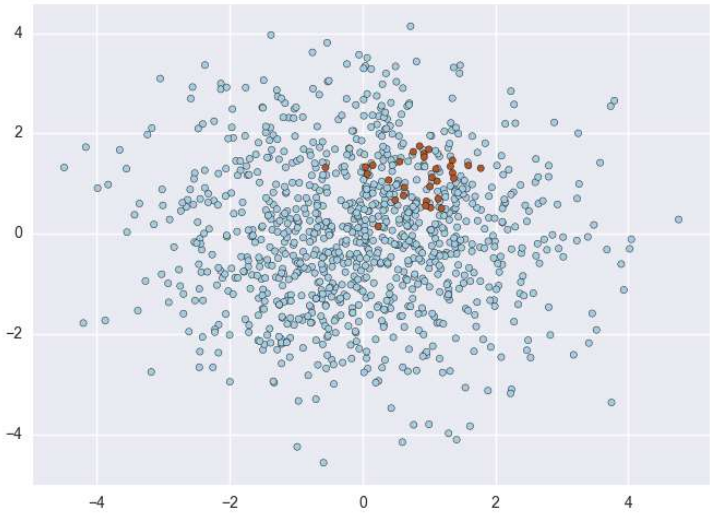
	A	B	C	D	E
0	NaN	2.0	0.0	NaN	9
1	NaN	0.0	NaN	1.0	5
2	NaN	4.0	4.0	NaN	3
3	NaN	NaN	NaN	5.0	8

NaNs in a dataset usually indicates:
- the data doesn’t exist

- the data does exist but we don't know what it is
Based on the specific use case, the appropriate measures should be taken.

Class Imbalance

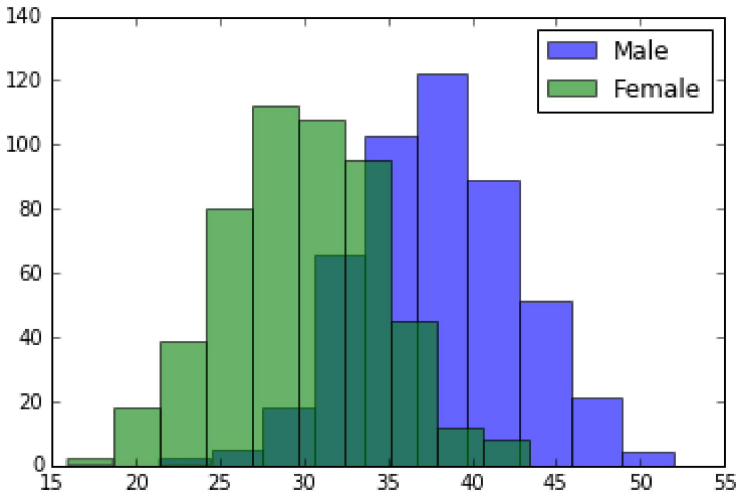
Specifically for supervised learning models, it is important for classes (or targets) to be balanced. However, in cases of fraud, it is very common to heave heavy class imbalance (e.g. only 2% of the dataset is actual fraud).



Such information is important to decide on the appropriate choices for feature engineering, modelling and model evaluation. For more information, check my blog on Fraud Detection Under Extreme Class Imbalance.

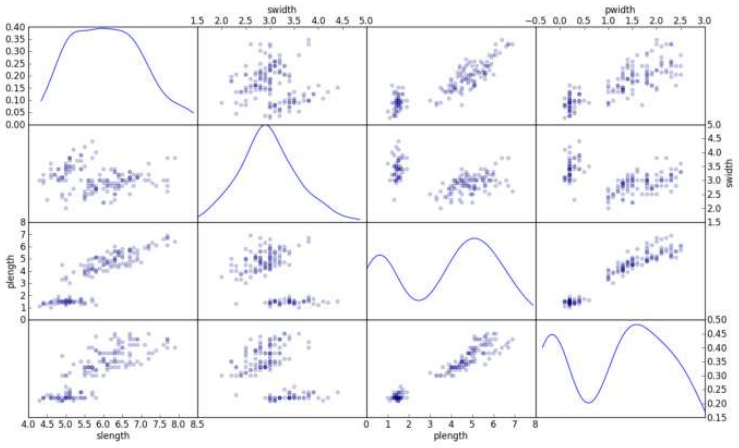
Univariate Analysis

Univariate analysis of single features (ignoring co-variate effects) is important when trying to look for outliers and unusual spikes in the variance. Common univariate analysis of choice is the histogram.



Bivariate Analysis

In bivariate analysis, each feature is compared to other features in the dataset. This would include correlation matrix, co-variance matrix or my personal favorite, the scatter matrix.



Scatter matrices allow us to find hidden patterns such as
- features that should be engineered together
- features that may need to be eliminated to avoid multicolinearity

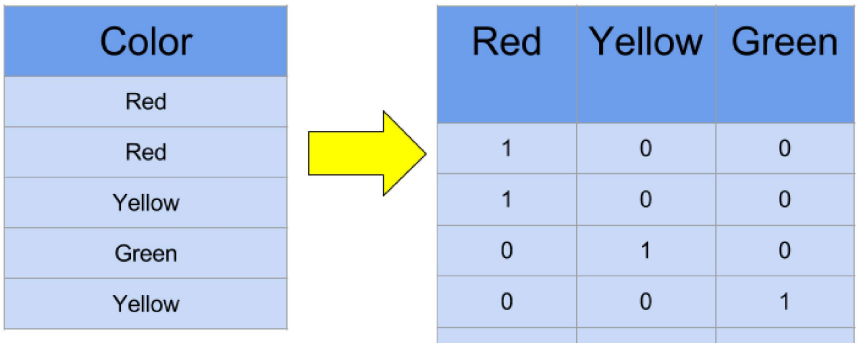
Multicollinearity is actually an issue for multiple models like linear regression and hence needs to be taken care of accordingly.

Feature Engineering

Once the data is collected, cleaned and analyzed, it's time to start creating features to be used in the model. In this section, we will explore some common feature engineering tactics.

Transformation

At times, the feature by itself may not provide useful information. For example, imagine using internet usage data. You will have YouTube users going as high as Giga Bytes while Facebook Messenger users use a couple of Mega Bytes. The simplest solution here would be to take the LOG of the values. Another issue is the use of categorical values. While categorical values are common in the data science world, realize computers can only comprehend numbers. In order for the categorical values to make mathematical sense, it needs to be transformed into something numeric. Typically for categorical values, it is common to perform a One Hot Encoding. In One Hot Encoding, a new feature is created for each categorical value to state if it is present in the given record. Example of One Hot Encoding is given below:



Color
Red
Red
Yellow
Green
Yellow

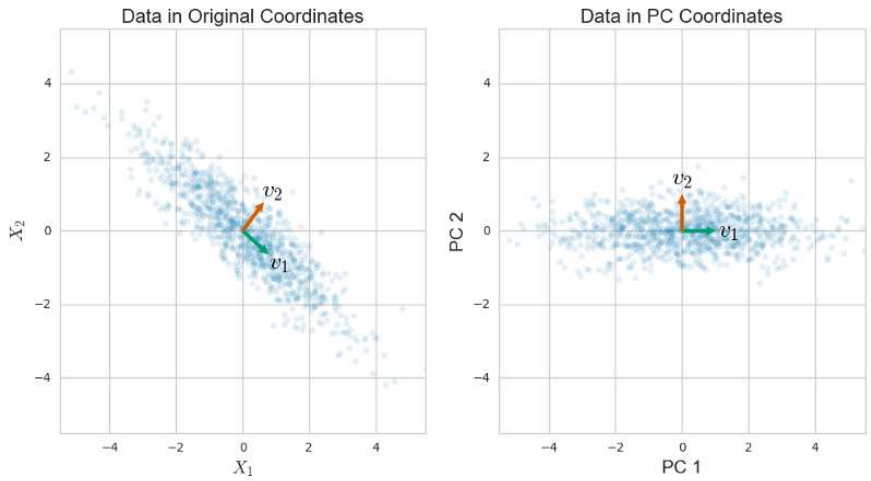
Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Combination

Certain features are redundant by themselves but are useful when grouped together. For example, imagine you had a predictive model for traffic density and you had a column for each type of car. Naturally, you don't care about the type of car but the frequency of the total number of cars. Hence, a row wise summation of all the car types can be done to create a new "all_cars" variable.

Dimensionality Reduction

At times, having too many sparse dimensions will hamper the performance of the model. For such situations (as commonly done in image recognition), dimensionality reduction algorithms are used.



An algorithm commonly used for dimensionality reduction is Principal Components Analysis or PCA. Learn the mechanics of PCA as it is also one of those topics amongst **COMMON INTERVIEW QUESTIONS**!!! For more information, check out my blog on The DOs and DON'Ts of Principal Component Analysis.

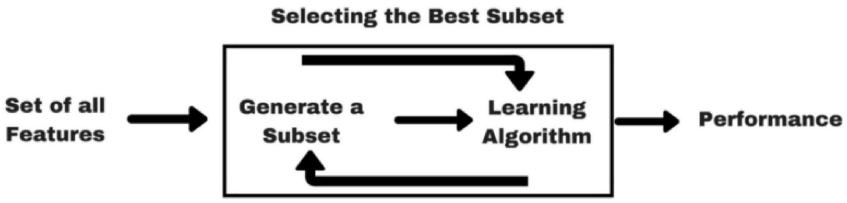
Feature Selection

Now that you have engineered your list of features, it is now time to select the features that will help build the most optimum model for the use case. The common categories and their sub categories are explained in this section.

Filter Methods

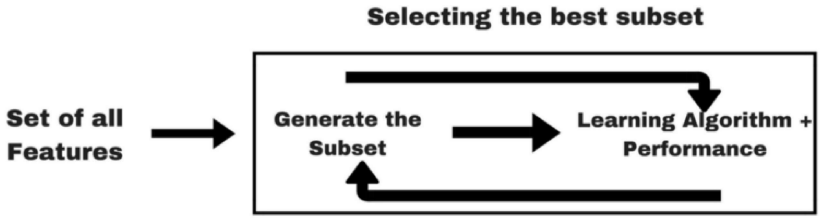
Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here. Common methods under this category are Pearson's Correlation, Linear Discriminant Analysis, ANOVA and Chi-Square.

Wrapper Methods



In wrapper methods, we try to use a subset of features and train a model using them. Based on the inferences that we draw from the previous model, we decide to add or remove features from your subset. The problem is essentially reduced to a search problem. These methods are usually computationally very expensive. Common methods under this category are Forward Selection, Backward Elimination and Recursive Feature Elimination.

Embedded Methods



Embedded methods combine the qualities' of filter and wrapper methods. It's implemented by algorithms that have their own built-in feature selection methods. LASSO and RIDGE are common ones. The regularizations are given in the equations below as reference:

Lasso:

Ridge:

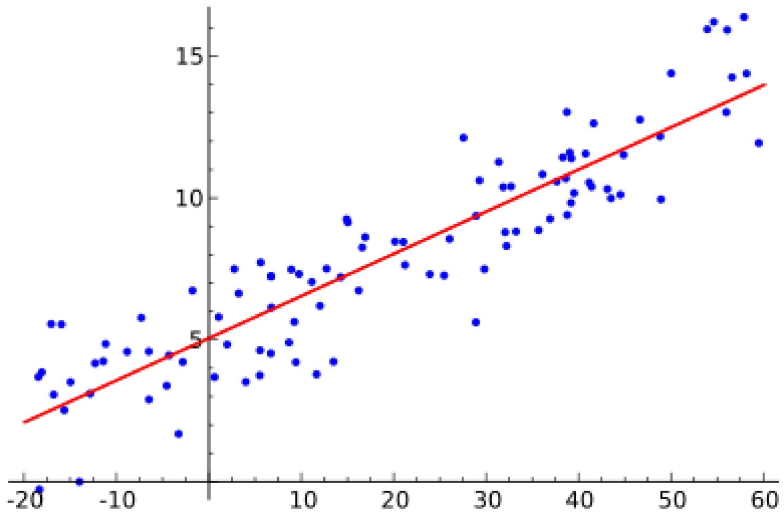
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

That being said, it is **VERY IMPORTANT** to understand the mechanics behind LASSO and RIDGE for interviews.

Machine Learning Models

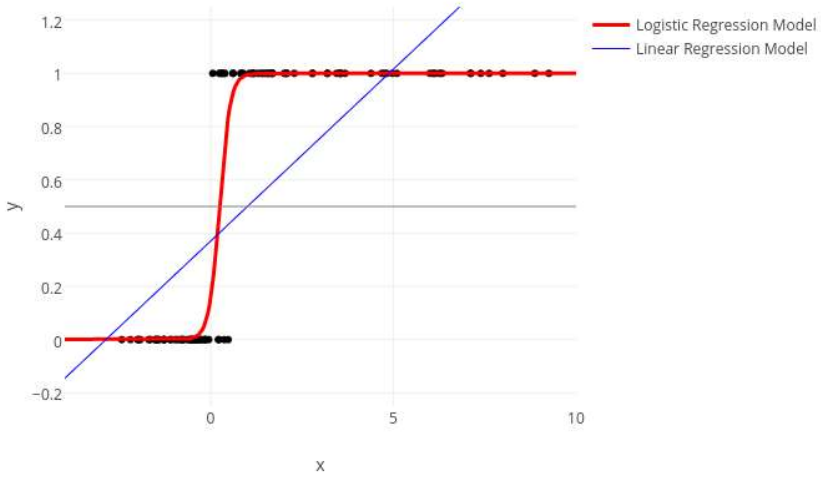
Now that we have our optimal features, it is now time to train our actual model! Machine Learning models fall into one of two camps: Supervised and Unsupervised. Supervised Learning is when the tags are available. Unsupervised Learning is when the tags are unavailable. Get it? SUPERVISE the tags! Pun intended. That being said, **DO NOT MIX UP THE DIFFERENCE BETWEEN SUPERVISED AND UNSUPERVISED LEARNING**!!! This mistake is enough for the interviewer to cancel the interview. Also, another noob mistake people make is not normalizing the features before running the model. While some models are resistant to this issue, a lot of models (like linear regression) is very sensitive to scaling. Hence. Rule of Thumb. **ALWAYS NORMALIZE THE FEATURES BEFORE USE!!!**

Linear and Logistic Regression



Linear and Logistic Regression are the most basic and commonly used Machine Learning algorithms out there. Before doing any analysis **MAKE SURE YOU DO LINEAR/LOGISTIC REGRESSION FIRST AS BENCHMARK!** One common interview blooper people make is starting their analysis with a more complex model like Neural Network. No doubt, Neural Network is highly accurate. However, benchmarks are important. If your simple regression model already has a 98% accuracy and really close to over-fitting, getting a more complex model is not a smart move. That being said, linear regression is used for continuous targets while logistic regression is used for binary targets (mainly

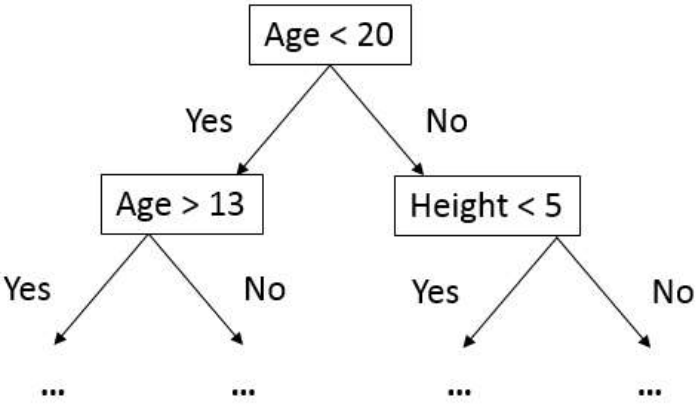
because the sigmoid curve forces the feature inputs towards either 0 or 1).



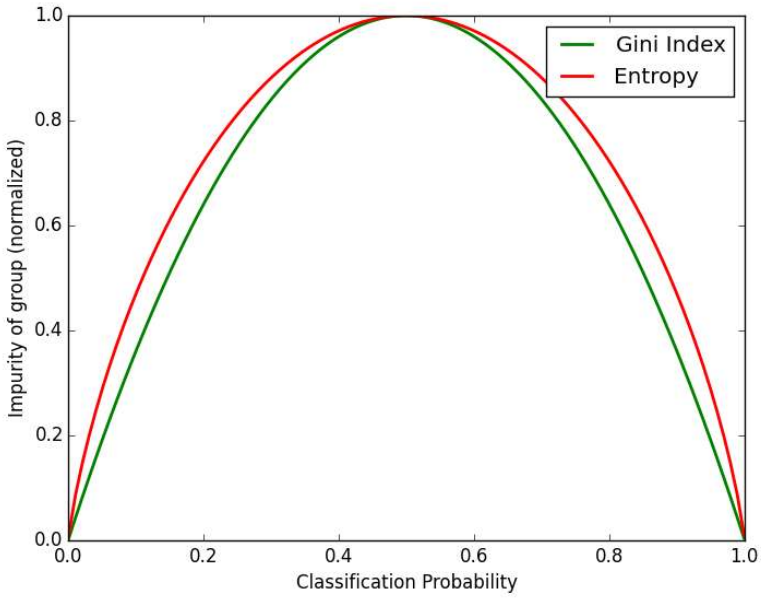
I would recommend the derivation of both logistic and linear regression (both single variate and multivariate). On top of preparing for the interview, the linear regression model is used as the base of a whole range of other machine learning models out there. Hence, it is long term investment.

Decision Trees and Random Forests

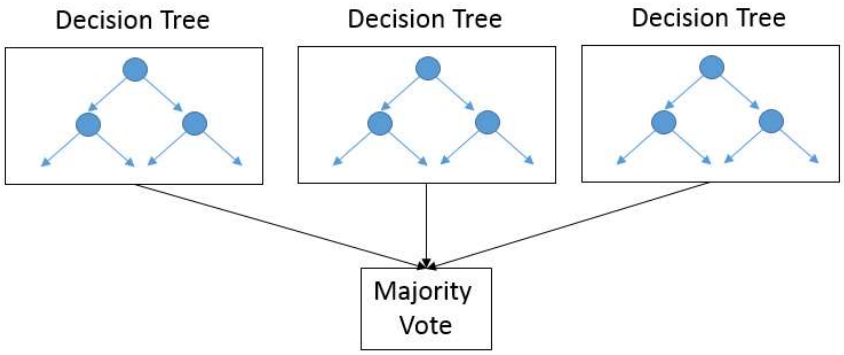
A slightly more complex model than a linear regression model is the decision tree. The decision tree algorithm splits at different feature based on information gain, until it hits a pure leaf (i.e. a set of records with only 1 label). A decision tree can be made to stop after a certain number of splits to stop it from getting pure leafs (common tactic to fix over-fitting problems).



The Information Gain calculated to split the tree is important. **COMMON INTERVIEW PROBLEM! ENSURE YOU KNOW HOW INFORMATION GAIN IS CALCULATED!!!** The common Information Gain calculation functions are Gini and Entropy.



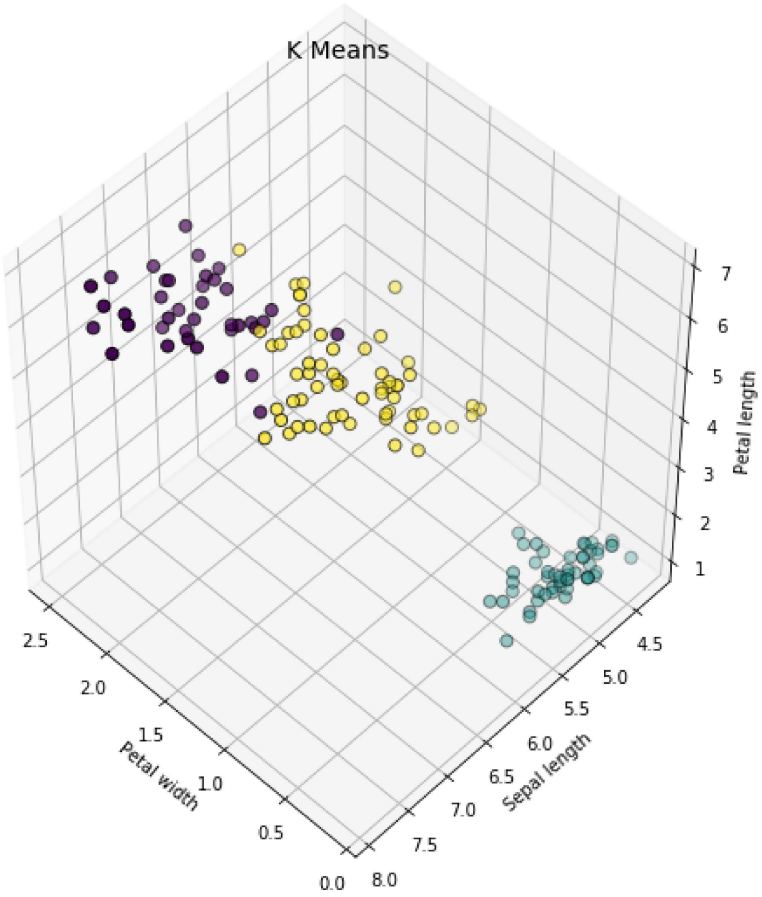
What is important in the above curve is that Entropy gives a higher value for Information Gain and hence cause more splitting compared to Gini.



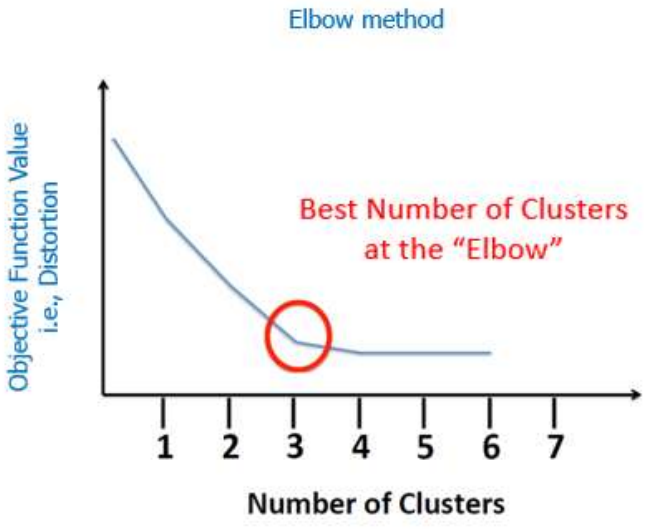
When a Decision Tree isn't complex enough, a Random Forest is generally used (which is nothing more than multiple Decision Trees being grown on a subset of the data and a final majority voting is done). Random Forest algorithms can over-fit if the number of trees are not determined properly. For more information on decision trees, random forest and tree based ensemble models, check out my other blog: Study of Decision Trees and Ensembles on Scikit-Learn

K-Means

K-Means is an unsupervised learning model that classifies data points into clusters. The number of clusters is provided, causing the model to shift the centroid until it iteratively finds the optimal cluster centers.



The number of clusters are determined using an elbow curve.

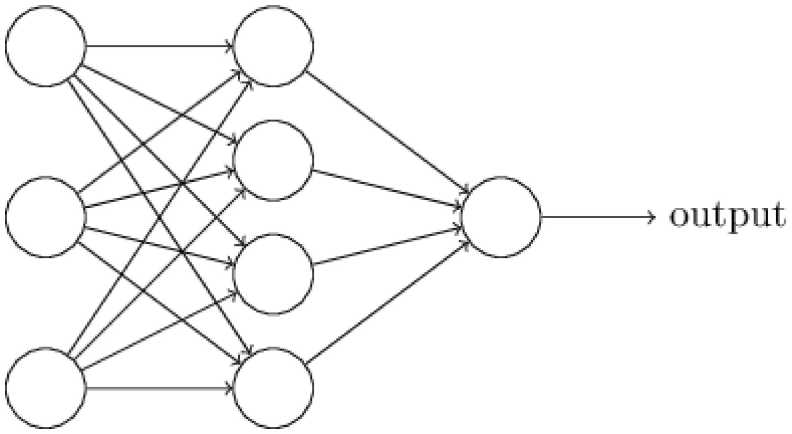


The number of clusters may or may not be easy to find (especially if there isn't a clear kink on the curve). Also, realize that the K-Means algorithm optimizes locally and not globally. This means that your clusters will depend on your initialization value. The most common initialization value is calculated in K-Means++, where the initial values are are far from eachother as possible. For more details on K-

Means and other forms of unsupervised learning algorithms, check out my other blog: Clustering Based Unsupervised Learning

Neural Network

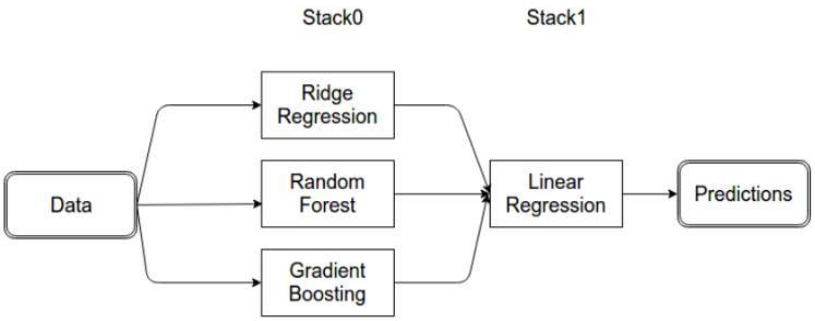
Neural Network is one of those buzz word algorithms that everyone is looking towards these days.



While it is not possible for me to cover the intricate details on this blog, it is important to know the basic mechanisms as well as the concept of back propagation and vanishing gradient. It is also important to realize that a Neural Network is essentially a black box. If the case study require you to build an interpretive model, either pick a different model or be prepared to explain how you will find how the weights are contributing to the final result (e.g. the visualization of hidden layers during image recognition).

Ensemble Models

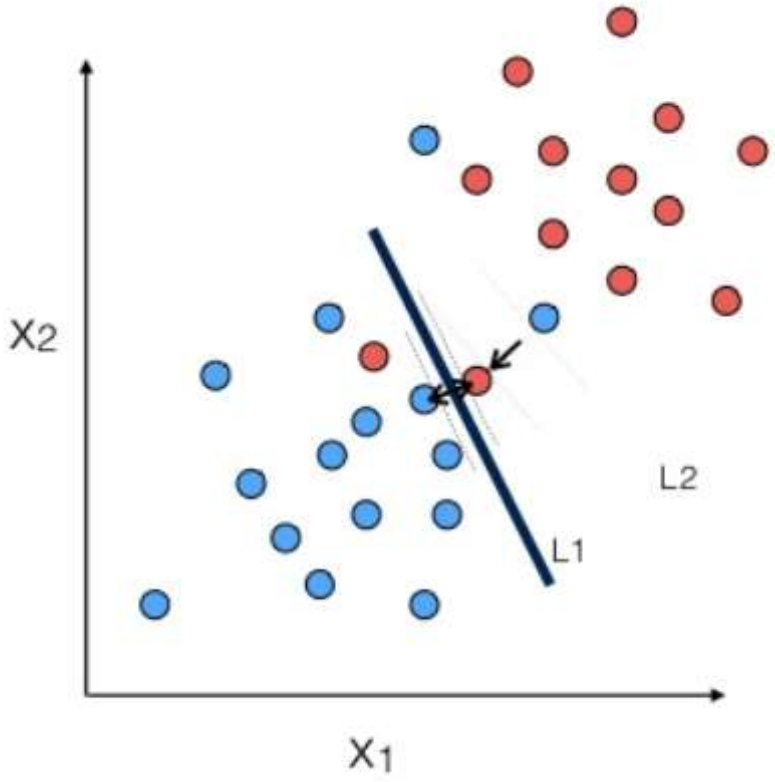
Finally, a single model may not accurately determine the target. Certain features will need special models. For such circumstances, an ensemble of multiple models are used. An example is given below:



Here, the models are in layers or stacks. The output of each layer is the input for the next layer.

Model Evaluation

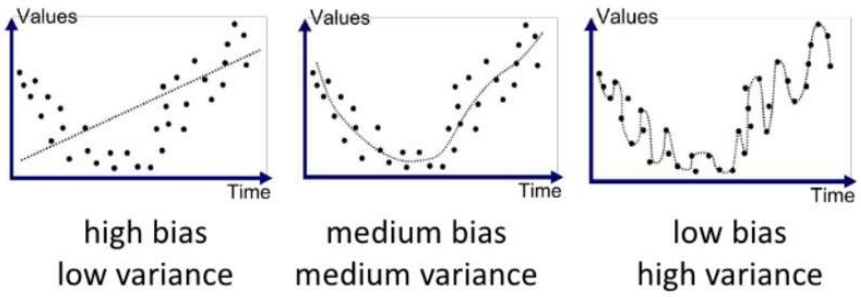
Classification Score



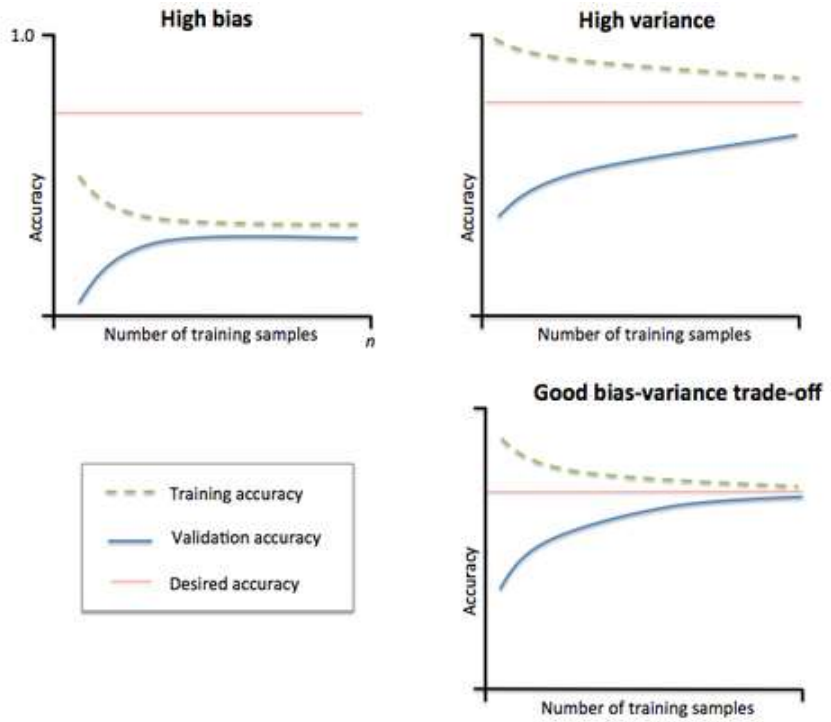
One of the most common way of evaluating model performance is by calculating the percentage of records whose records were predicted accurately.

Learning Curve

Learning Curve is also a common method for evaluating models. Here, we are looking to see if our model is too complex or not complex enough.



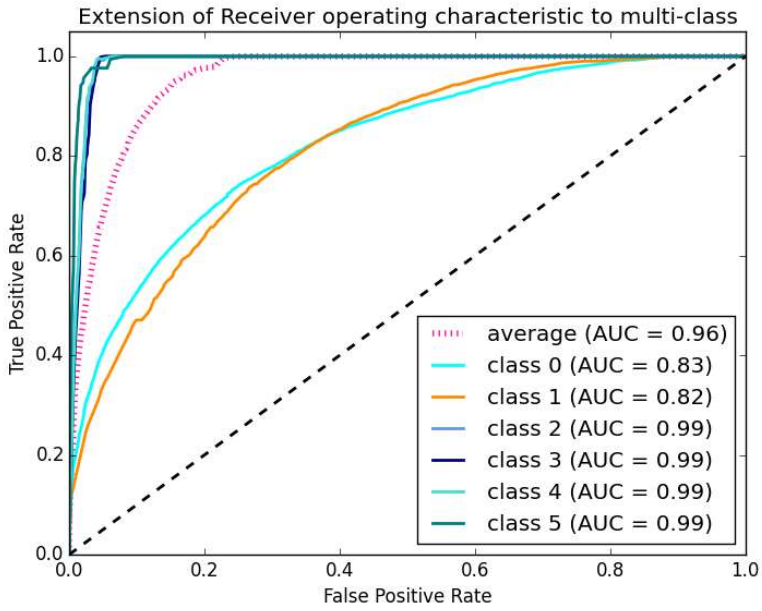
If the model is not complex enough (e.g. we decided to use a linear regression when the pattern is not linear), we end up with high bias and low variance. When our model is too complex (e.g. we decided to use a deep neural network for a simple problem), we end up with low bias and high variance. High variance because the result will VARY as we randomize the training data (i.e. the model is now very stable). **DO NOT MIX UP THE DIFFERENCE BETWEEN BIAS AND VARIANCE DURING THE INTERVIEW!!!** Now, in order to determine the model's complexity, we use a learning curve as shown below:



On the learning curve, we vary the train-test split on the x-axis and calculate the accuracy of the model on the training and validation datasets. If the gap between them is too wide, it's too complex (i.e. over-fitting). If neither one of the curves is hitting the desired accuracy and the gap between the curves is too small, the dataset is highly biased.

ROC

When dealing with fraud datasets with heavy class imbalance, a classification score does not make much sense. Instead, Receiver Operating Characteristic or ROC curves offer a better alternative.



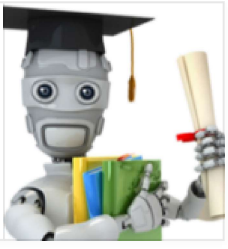
The 45 degree line is the random line, where the Area Under the Curve or AUC is 0.5 . The further the curve from this line, the higher the AUC and better the model. The highest a model can get is an AUC of 1, where the curve forms a right angled triangle. The ROC curve can also help debug a model. For example, if the bottom left corner of the curve is closer to the random line, it implies that the model is misclassifying at Y=0. Whereas, if it is random on the top right, it implies the errors are occurring at Y=1. Also, if there are spikes on the curve (as opposed to being smooth), it implies the model is not stable. When dealing with fraud models, ROC is your best friend.

Additional Materials

Stanford Machine Learning | Coursera

About this course: Machine learning is the science of getting computers to act without being...

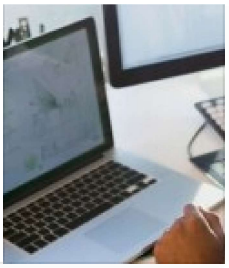
www.coursera.org



University of Washington Machine Learning Specialization | Coursera

This Specialization from leading researchers at the University of Washington introduces you to the...

www.coursera.org



Deep Learning Specialization | Coursera

Deep Learning from deeplearning.ai. If you want to break into AI, this Specialization will help you do...

www.coursera.org