ML Interview Practice December 24, 2016

1. We A/B tested two styles for a sign-up button on our company's product page. **100** visitors viewed page **A**, out of which **20** clicked on the button; whereas, **70** visitors viewed page **B**, and only **15** of them clicked on the button. Can you confidently say that page **A** is a better choice, or page **B**? Why?

Answer:

First, I would like to capture and repeat the information given. You are testing two variations of the sign-up button on the company's page, and you have gathered some statistics on Page A and Page B as below:

Page A: 20 clicked out of 100 viewed

Page B: 15 clicked out of 70 viewed

Although on the surface, it may seem that the highest absolute number of clicks which is Page A, however, this may be misleading as the number of people who viewed Page A and Page B are not equal, as Page A has 100 views while Page B has only 70 views. For a comparison to be valid between the two, a normalization should be done. One simple method is to divide the number of clicks for that particular page by the number of views for that page. In effect, this yields the click-per-view ratio or percentage for that page.

This would mean that if we simplify the given ratios:

$$Page\ A = \ P_A = \frac{20\ clicks}{100\ views} = \frac{1}{5} = 0.2\ clicks\ per\ view$$

$$Page\ B = P_B = \frac{15\ clicks}{70\ views} = \frac{3}{14} = 0.214\ clicks\ per\ view$$

From the above, it is now clear that Page B has a slight edge over Page A, with 0.214 clicks per view versus 0.2 clicks per view, even though Page A has a higher absolute number of clicks. This shows the dangers of misleading results if it is wrongly interpreted and presented.

To verify whether Page B is indeed more statistically significant than Page A, we would need to compute the Z-statistic of the distribution. For a sufficiently large population, the Central Limit Theorem applies such that the resulting distribution can be considered as near normal or normal. Defining the null hypothesis and alternate hypothesis is as below:

Null hypothesis = $P_A = 0.2$

Alternate hypothesis = $P_B = 0.214$

The Z-score will tell whether to reject or not the null hypothesis. It returns a value between -3 and +3, where it is interpreted as "between -3 standard deviations away and +3 standard deviations away from the mean". The p-score probability (values between 0 – 1) is also another commonly calculated statistic alongside the Z-score, where a small p-value (typically 0.05 or less) corresponds to a strong evidence against the null hypothesis, so that the null hypothesis may be rejected. For example, for a 95%

confidence level to reject the hypothesis, the Z-score will be either -1.96 or +1.96 standard deviations, and the p-value will be 0.05.

2. Can you devise a scheme to group Twitter users by looking only at their tweets? No demographic, geographic or other identifying information is available to you, just the messages they've posted, in plain text, and a timestamp for each message.
   In JSON format, they look like this:

```
{
 "user_id": 3,
 "timestamp": "2016-03-22_11-31-20",
 "tweet": "It's #dinner-time!"
}
```

   Assuming you have a stream of these tweets coming in, describe the process of collecting and analyzing them, what transformations/algorithms you would apply, how you would train and test your model, and present the results.

Answer:

The task is to analyze and classify a stream of tweets into a few groups of people with a characteristic tweeting behavior. My input will be tweets in the form of JSON, which may contain information such as the User ID, timestamp and the raw tweet text. And the output will be a classification of a particular User ID to a group of users. Since the groups are unknown and unlabeled, this can be considered as an unsupervised classification problem.

The tweets can be collected using an API from Twitter. Since the input is in JSON form, we would require a transformation of the data into a form that is readily able to be processed by a machine learning algorithm. An example would be a structured table where each feature is represented as a column and each row a unique User ID.

User ID	Timestamp	Tweet
3	2016-03-22_11-31-20	It's #dinner-time!

One way to preprocess the tweets before feeding them into a machine learning algorithm is to remove punctuations, common verbs and adjectives to reduce tweets to only stem or root words: those which will give most context to the content of a particular tweet. It is a form of Natural Language Processing. Also, hashtags can be extracted alongside stem words as terms or keywords which the machine learning algorithm can then fit to, such as a term-document matrix or a similarity matrix. I would pick a few tweets at random and hypothesize the group types it would belong to, whether it is by subject (world, politics, sports, economy, entertainment) or sentiment (positive, negative, neutral) to name a few.

I would start off with using K-Means as the algorithm of choice, simply because it is a widely-used algorithm for unsupervised clustering and it is generally a good starting algorithm due to its simplicity to understand and implement. I would train the collated dataset with K-Means, then validate its performance using the silhouette score against the prediction of a few sample points taken from the training set. I would try out a range of cluster values for example, from 2 to 10, and determine the best

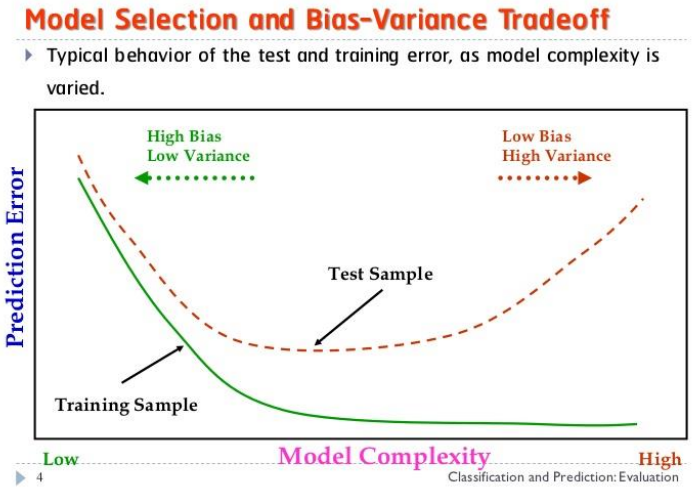
number of clusters to be set based on the score. I would present the results as users who utilize a group of certain words or hashtags for a particular cluster. An example is as below. To allow easier interpretation, I would also provide some data visualization charts and graphs.

Cluster	Words	Entertainment	Politics	Sports
Cluster 1	Win, game, match	70	20	300
Cluster 2	…	…	…	…
Cluster 3	…	…	…	…

3. In a classification setting, given a dataset of labeled examples and a machine learning model you're trying to fit, describe a strategy to detect and prevent overfitting.

Answer:

To detect the occurrence of overfitting, a few tools can be used. One of them is by plotting a learning curve, where the y-axis is the model's performance in terms of accuracy or any other performance metric and the x-axis is the number of training examples. The model is trained and evaluated on an increasing number of training example batches. Two graphs are plotted: a training graph and testing graph. If the two graphs are close to each other, it means that the model is underfitting due to error in bias, while if the curves start to diverge from one another, then the model is starting to overfit due to error in variance. This inflection point would be a good indicator of overfitting. To reduce the risk of overfitting, I can choose a number of options. One is to increase the training examples so that the model is able to learn more on the underlying features describing the dataset. The other option is to reduce the model's complexity by reducing the order of a polynomial model for example. By reducing the model's complexity, the model would not attempt to capture every nuance in the training data and consequently fail to generalize to unseen data which contributes to the overfitting issue. Depending on which resource is available and the goals of performance expected from the model, I may need to make tradeoffs.



4. Your team is designing the next generation user experience for your flagship 3D modeling tool. Specifically, you have been tasked with implementing a smart context menu that learns from a modeler's usage of menu options and shows the ones that would be most beneficial. E.g. I often use **Edit** > **Surface** > **Smooth Surface**, and wish I could just right click and there would be a **Smooth Surface** option just like **Cut**, **Copy** and **Paste**. Note that not all commands make sense in all contexts, for instance I need to have a surface selected to smooth it. How would you go about designing a learning system/agent to enable this behavior?

Answer:

I would strategize a plan firstly on data input, then data cleaning/transformations, model selection, training, evaluation and validation, and finally reporting and fine tuning. For data input, I would need to define an outcome of this design task, which is to implement a smart context menu based on the user's history of the menu item's usage. With that in mind, I would also need to decide whether a supervised learning or unsupervised learning approach would be the most suitable for the task, and whether is this a classification or regression exercise. Since the outcome would be to predict which appropriate menu item should be displayed in the smart menu, this would fare well in a supervised classification task. Then, I need to define an appropriate set of features and labels for the machine learning algorithm which would capture the most out of the underlying reasons for a good prediction from the user's point of view. For example, showing a most-used menu item of the user would be a good starting point, as users would benefit from this software feature. Thus, this can be included as part of the items a machine learning algorithm should consider. An example feature/label table is as below:

Features					Label
Menu item selected	No. of times menu path is selected / session	Previous user action before menu selection	Previous menu item selected	Next menu item selected	Include in Smart Menu?
Smooth Surface	20	Surface selected	Move Surface	Smooth Surface	Yes / No

In addition, if necessary, the data would then be normalized or standardized so as to not cause any one particular feature to have a major influence on the outcome of the prediction, as misguided by the machine learning algorithm. When choosing an algorithm, I would start with a suitable classification algorithm with generally good performance, such as Random Forest or Support Vector Machines, only then I will begin to modify the model's complexity and the chosen algorithm based on evaluation and validation of the model.

5. Give an example of a situation where regularization is necessary for learning a good model. How about one where regularization doesn't make sense?

Answer:

Regularization is a common technique used to reduce overfitting. One example is when modelling the house price based on observed features. Let's say a model is implemented on a dataset which has the number of rooms, square footage of the house, and neighbourhood safety level. Upon training and testing the model, you have discovered that the model is doing fairly well, but not as perfect as you

wish. So you have decided to add more features into the dataset for training. (for example: education level, distance to public transport, garage size, traffic volume, noise levels, etc). After feeding in the new dataset, it turns out that the model is performing much better in training. But with all that added features, the model has relied too much on the training data, and thus has overfit and will fail to perform well on generalizing unseen data. Regularization comes into play as a technique that penalizes the model for trying to fit too much to the data. An example would be having an assistive steering for your car. While turning a corner, you will have total control of the steering, but you would probably have a lot of jerky steering movements while doing so. Assistive steering helps to smoothen the steer input to the vehicle so that a smooth turning is maintained. Thus, the overfitting-prone tendencies of the model is curbed by regularization and you get a good predictive performance for unseen data.

An example where regularization would not make sense, is when your model is already simple to begin with. Taking the house price prediction as an example again, this time, you have chosen only the square footage of the house as the feature. Since there is not much information for the algorithm to learn on, the predictive power will not be good, and adding regularization would be useless as the model is not suffering from overfitting but an error due to bias because of its low complexity being unable to capture the underlying features of the data. In this case, the better option is to gather more training data with more features or increase the model's complexity.

6. Your neighborhood grocery store would like to give targeted coupons to its customers, ones that are likely to be useful to them. Given that you can access the purchase history of each customer and catalog of store items, how would you design a system that suggests which coupons they should be given? Can you measure how well the system is performing?

Answer:

Given that the dataset inputs come from the customer's purchase history and the catalog of store items, the store wants to give coupons to customers which are useful to them.

Since we would want to investigate which coupon would be most attractive to a certain group of customers, a supervised learning algorithm approach would be a good candidate. Since there are no other customer identifying information that can be gathered, such as age, gender or income, to measure the success of the coupon's campaign, we will need to rely on various aspects of the products purchased by the customer. For the data collection portion, a table of the customer's purchase amount of the types of products sold by the grocery, the frequency of purchase of products and the price tier of the products can be collated into a data table. We can also investigate the behavior of customer who purchase items on sale and whether they too will redeem coupons. A sample table can be seen below:

Product Category 1 (kg/month)	Product category 2 (kg/month)	Product category 1 (transactions/month)	Product category 2 (transactions/month	Product 1 Price Tier	Product 2 Price Tier	Product Category 1 Total Discount (%)	Product Category 2 Total Discount (%)	Coupon redeemed
500	1000	600	200	1	2	10	20	Coupon A
3000	600	700	150	1	1	10	0	Coupon B
100	200	50	75	3	2	0	0	Coupon C

This table would allow a machine learning algorithm to investigate some underlying reasons behind a successful coupon redemption campaign. To measure how well the system is performing, the acceptance/redemption rate of a particular coupon can be considered as a performance metric. An algorithm that predicts given coupons which have the highest redemption rates would mean that the coupons are correctly targeted to the customer's desires. For example, for the base case without the system's intervention, there are three coupons (A, B, and C) of 10,000 pieces each which are distributed equally to customers at random. We can then measure the base performance of the coupons by tallying how many of Coupon A/B/C is redeemed on a weekly/monthly average. Then, upon implementation of the system, it will recommend which coupon to give to a customer upon checkout. The number of coupons redeemed are tallied again to check if the system's recommendation did improve the acceptance rate of the coupons upon the base case.

7. If you were hired for your machine learning position starting today, how do you see your role evolving over the next year? What are your long-term career goals, and how does this position help you achieve them?

Answer:

For the next year, I would see myself being passionately curious to learn and understand the inner workings of the company, the industry as a whole, and how can I improve any aspect of it with machine learning as a primary tool. My long term goals for my career are to be part of a benefitting cause to the advancement of research or betterment of human life, thus, a role where I could work with people from all sorts of discipline and expertise that cares for the same cause as I am who are brilliant ideators. This position will help me strengthen my skills in machine learning to be able to build high impact solutions in the near future.