ML Interview Practice
Christopher Jose

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?**

For this problem, hypothesis testing of the difference of two population proportions is appropriate, given that we are concerned as to whether the proportion of page A views resulting in a button click is greater than the proportion of page B views resulting in a button click. Since there are at least 5 observations for each of button clickers and non-clickers within both samples, the central limit theorem applies. I will conduct a two-proportion z-test.

Null Hypothesis: page A conversion rate = page B conversion rate
Alternative Hypothesis: page A conversion rate != page B conversion rate

Code is shown below for conducting a hypothesis test to assess if page A has a different conversion rate from page B. We are 95% confident that the population page A conversion rate is between -28.6 and 25.7 percentage points greater than the population page B conversion rate. This means that we are not confident that page A is the better choice. In fact, page A could be a worse choice, given the results of this hypothesis test. There is a 91.8% likelihood of observing the observed difference in sample proportions or a difference more extreme assuming the difference in population proportions is zero. This metric indicates our results are likely assuming the proportions are the same.

Suppose page A is Educational Testing Service's (ETS) homepage, but with the various testing products (GRE, HiSET, etc) highlighted in a certain color and with a bigger font. Suppose page B had regular font sizes and no colors on the testing product buttons. Then we are not confident that these two home pages for ETS are statistically significantly different.

```
n_b=15
n_a=20
p_b=15/70.0
p_a=20/100.0
print(p_b,p_a) #A probability, B probability
```

```
0.21428571428571427 0.2
```

```
from scipy import stats
p_avg=(n_a*p_a+n_b*p_b)/(n_a+n_b)
standard_error=np.sqrt((p_avg*(1.0-p_avg))*((1.0/n_b)+(1.0/n_a)))
z=((p_a-p_b)-0)/standard_error
margin_of_error=1.96*np.sqrt((p_b*(1.0-p_b)/n_b)+(p_a*(1.0-p_a)/(n_a)))
conf_lower=(p_a-p_b)-margin_of_error
conf_upper=(p_a-p_b)+margin_of_error
p_value=(stats.norm.cdf(z))*2 #z is negative, so we double P(Z<z) to include both tails
print ('z: %.2f \n\
margin of error: %.3f \n\
95%% confidence interval: (%.3f, %.3f) \n\
p value: %.5f' %(z,margin_of_error,conf_lower,conf_upper,p_value))
```

```
z: -0.10
margin of error: 0.272
95% confidence interval: (-0.286, 0.257)
p value: 0.91765
```

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.

I would load the tweets into Python as a pandas DataFrame. Each tweet would have its own row. Dictionary keys would become column names, and dictionary values would become row values. I would convert the timestamp string to a datetime format so that different parts of the string could be easily extracted and pandas datetime operations could be performed. I would make a field for time, which would effectively provide location information, since users of certain countries mainly tweet at certain times.

I would then tokenize and transform the dataset by splitting the "tweet" column into many columns. Each column would be a word or phrase from all the tweet data. Each column's value in each row would be the number of times that the word/phrase appeared in the tweet. For example, tweets from ETS's twitter account would result in columns containing frequencies for words such as "teacher", "education", etc. Filler words with little meaning would be removed, like "the", "is", etc.

I would then standardize the data and use clustering techniques, such as hierarchical clustering to bucket the data into groups. I would try to interpret each group. Using these group numbers, I would evaluate multiple classifiers on the data via cross validation and select the model with the best cross validation score. Then, I would train this model off all the data and use it to predict the group for each future tweet fed to it.

I would present results that include: interpretations for each group, models and their evaluation metrics, the selected model, and how often to consider remaking the groups and rebuilding the models.

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.

I would plot validation curves. The y-axis would show an evaluation metric, like F1 Score. The x-axis would show hyperparameter values or number of training examples. I would show two curves. One curve would be associated with the train data. The other curve would be associated with the test data. I would draw conclusions from comparing the two curves. As x increases, if the train score improves a lot, while the test score does not, then this would indicate high model flexibility, high variance, and over-fitting. In this situation, the model captures noise in the

training data. It fits to the training data well. However, this noise is not useful in predicting other data. The lower test score reflects this. I would not use hyperparameter values where the test score starts staying level (or gets worse), while the training score starts getting much higher.

There are multiple methods that can be used to avoid over-fitting. Regularization techniques like Lasso Regression reduces over-fitting by performing feature selection and reducing the feature space. Ridge Regression is like the Lasso, but it does not do feature selection. The idea behind these two methods is reducing model variance by an amount that is less than the resultant increase in model bias. In general, regularization methods penalize models for being overly complex.

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?

The reinforcement q-learning algorithm would be a good choice for this problem.
I would define states and actions for the learner. I would set up a Q-table containing (state, action) pairs and their associated Q-values. The Q-value would be based on the rewards received for the (state, action) pair. I would consider the context menu as consisting of "option slots". Each slot can have an option in it. There are a fixed number of option slots (N).

When the learner performs an action, it will consist of the selection of an option to add to the option menu. The N options with the highest Q values are added to the option menu. If there are less than N options, then just these options are added to the option menu. If there is a tie in Q-values greater than zero, then one of these options is selected randomly. States with zero Q-values are never added to the option menu. Zero Q-values would indicate that the user never selected the option before, which could also indicate that the option is not possible in that section of the tool.

States will consist of features giving context details. These details allow the learner to figure out whether it's possible for the user to use the given option when they right click. Options that are possible in a certain state will have positive reward in this state. If an option can't be selected in a certain part of the software, then the Q-value for this (state, action) pair will never increase above zero. Q values will be updated according to a value iteration update rule. This means that Q won't just have its value change by the amount of the reward, but by an amount proportional to the reward amount. When the user selects an option in a certain section of the tool, then the Q-value for this (state, action) pairing will increase by an amount that is a function of the reward. Thus, over time, if an option is repeatedly selected in a state, then its Q value will keep increasing. Additionally, Q-values for options in a certain state that used to get selected a lot, but are no longer getting selected, should begin decreasing as those options start not getting selected

and other options start getting selected in that state. This is to ensure that the algorithm is flexible to a change in the user's habits.

I will include a decaying ϵ (exploration) variable which decreases towards zero as the q-learner updates itself many times. A random action will be selected with ϵ probability, regardless of the Q-values. This will help the learner avoid getting stuck at local minimums.

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?

Suppose ETS is predicting GRE scores based on a variety of variables, such as age, height, location, etc. If the model is incredibly flexible and fits to the data perfectly, then it will capture a lot of noise, have high variance, and will over-fit the data. Predicting new student GRE scores will be difficult because the model was built off not just patterns between GRE score and student demographics, but also superfluous information. Regularization will reduce this over-fitting by penalizing the model for being overly complex. The model's variance will decrease as the amount of penalization increases. It will eventually be able to generalize better to unseen data, since you'll be removing some of the noise. GRE scores that were not used in training the model will be better predicted. For example, Lasso Regression reduces over-fitting by performing feature selection and reducing the feature space.

Suppose ETS's model generalizes well to unseen data and has low variance. By adding regularization into the mix, ETS won't reduce much noise, since there isn't much noise to reduce. ETS will make the model less complex and increase the bias, when the model is already somewhat biased because of having low variance. Thus, the model won't capture the underlying patterns between GRE score and student demographics as well and won't perform as well on unseen student data.

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?

I would break this problem into an unsupervised learning problem and a supervised learning problem. For the unsupervised learning problem, I would use a clustering technique, such as hierarchical clustering, to assign customers to groups. This would be based on features indicating which products customers ordered in the past. Each row would be a different customer in the dataset. These clusters would be investigated and would hopefully reveal different customer types. For example, perhaps one customer grouping might represent customers who buy meat and dairy products.

Once cluster numbers are assigned to each observation, then the supervised learning problem is considered. A multi-level classification method would be trained off the dataset to predict the cluster for each customer. If a cluster represents customers who buy meat and dairy, then coupons containing meat and dairy items should be used for these customers.

The customer dataset can then be updated as new customers are added to it and existing customers make purchases. As this happens, clusters for new / modified observations can be predicted using the already trained classification model. Periodically, this model should be re-trained on the dataset, and the clustering algorithm should be rerun.

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?

I see my role as evolving from doing more descriptive analysis, meaning "what is happening", into more diagnostic analysis, meaning "why is it happening" (source 1). I would think that through exposure to the different processes at Educational Testing Services, such as supporting statistical operations and data processing, I will develop a lot of business-specific knowledge over time that will allow me to provide value in new ways. Over the long run, I hope to also get to do prescriptive analysis, meaning having a decision in "what needs to be done", so that I can help improve ETS's mission of measuring student learning. I would find it inspiring to help improve the foundation upon which great minds are being built. It all starts with education and its proper assessment.

Sources

(1) https://halobi.com/blog/descriptive-predictive-and-prescriptive-analytics-explained/