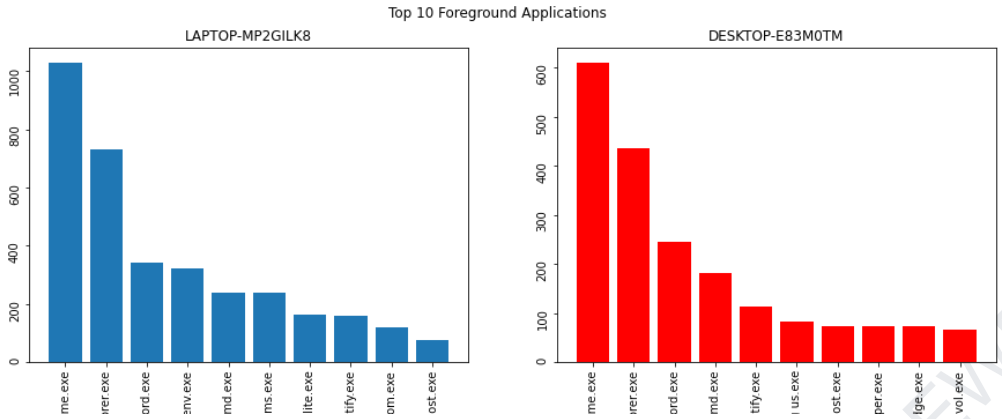
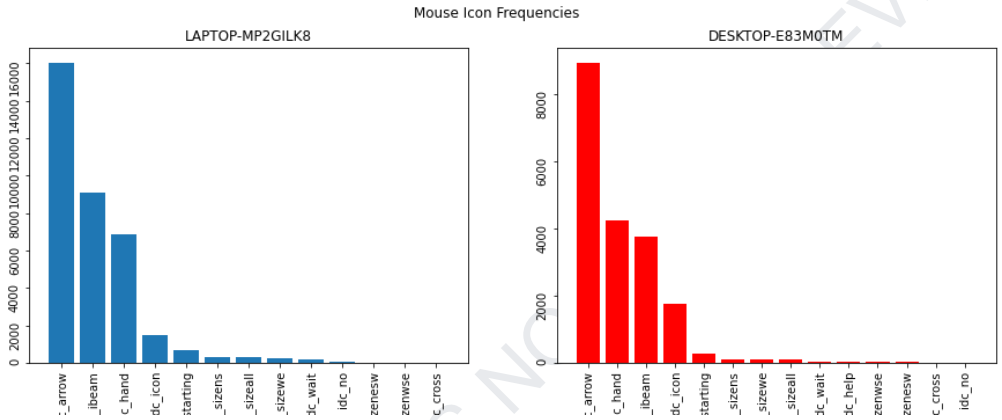
DSC180B B14 - Group 3 - Project

Hiroki Hoshida, Jared Thach, Cyril Gorlla

For Quarter 2 of our DSC180 project, we are attempting to solve 2 problems using the data we collected during Quarter 1. The first is to predict the next application the user will open based on their previous app history. The second is to predict the total amount of time a given application will be used based on previously collected data. We started work on Problem 1 first, and are in the exploratory data analysis stage of Problem 2.

In order to start our exploration of the datasets, we first combined all the database files collected and condensed them into a single sequential DataFrame ordered by timestamp. This single DataFrame could be filtered out further into separate collectors. For Problem 1, we extracted the app history dataset, which shows which app was opened when, and the cursor history dataset, which shows what shape the cursor icon was at a certain time. Using the app dataset, we first created a total appearance count of each application. We also created various visuals to help us better understand the data. For example, we graphed the most frequently occurring apps for different time periods, which showed us that the most used apps differed during the academic year vs. during break.





Out[5]:

chrome.exe	1028
explorer.exe	730
discord.exe	342
devenv.exe	320
cmd.exe	240
teams.exe	239
db browser for sqlite.exe	164
spotify.exe	157
zoom.exe	119
shellexperiencehost.exe	74
systemsettings.exe	65
netflix.exe	61
vsdebugconsole.exe	50
searchapp.exe	45
sndvol.exe	32
msedge.exe	30
microsoft.aad.brokerplugin.exe	23

After a general understanding of the dataset we had, we created functions to help us calculate the conditional probabilities of each application. We wrote three functions: `get_totals`, `get_cond_probs`, and `get_cond_probs_plots`.

`get_totals` and `get_cond_probs` work together to calculate the conditional probabilities of each application and create a matrix, and `get_cond_probs_plots` to generate plots showing the probabilities for each application in the dataset.

	anaconda3-2021.11-windows-x86_64.exe	applicationframehost.exe	chrome.exe	cmd.exe	db browser for sqlite.exe	devenv.exe	discor
anaconda3-2021.11-windows-x86_64.exe	0.000000	0.0	0.500000	0.500000	0.000000	0.000000	0.00
applicationframehost.exe	0.000000	0.0	0.500000	0.000000	0.000000	0.000000	0.50
chrome.exe	0.002457	0.0	0.000000	0.078624	0.028256	0.079853	0.17
cmd.exe	0.005495	0.0	0.510989	0.000000	0.010989	0.032967	0.06
db browser for sqlite.exe	0.000000	0.0	0.135338	0.015038	0.000000	0.278195	0.03
devenv.exe	0.000000	0.0	0.240157	0.011811	0.082677	0.000000	0.03
discord.exe	0.000000	0.0	0.553506	0.029520	0.014760	0.014760	0.00
easeofaccessdialog.exe	0.000000	0.0	1.000000	0.000000	0.000000	0.000000	0.00
explorer.exe	0.001684	0.0	0.336700	0.156566	0.112795	0.079125	0.04
githubdesktop.exe	0.000000	0.0	0.142857	0.000000	0.000000	0.000000	0.00

Using the conditional probabilities matrix, we could start developing our first order Hidden Markov model. We created a HMM class with attributes to store basic information on the model, such as the data, data count, priors, and posteriors. We also created a fit function that takes in training data and outputs prior and posterior probabilities using the previous `get_totals` and `get_cond_probs` functions. The fit function thus calculates the frequency of each unique pair of applications then stores their conditional probabilities (e.g. `P(chrome.exe|explorer.exe)`) into self.posteriors. We then created a predict function that predicts n applications that could follow an application, n being a parameter of the function. Thus, n indicates the number of predicted foregrounds per input observation. Finally, the accuracy function was created to determine the accuracy of the prediction by comparing the output of predict with actual values from the dataset. We based our function names and class structures to similar model APIs from sklearn, such as the fit and predict functions.

```
Out[17]: [['chrome.exe', 'cmd.exe', 'db browser for sqlite.exe'],
          ['chrome.exe', 'explorer.exe', 'zoom.exe'],
          ['chrome.exe', 'explorer.exe', 'zoom.exe'],
          ['chrome.exe', 'cmd.exe', 'db browser for sqlite.exe'],
          ['chrome.exe', 'cmd.exe', 'db browser for sqlite.exe']]
```

We also started work on Problem 2, to determine how long a given application will be used in total. In the exploratory data analysis conducted for Problem 2, we used the isolated datasets we created for Problem 1, From the data, we used the timestamps from the dataset to find the time lengths of each time the application was in the foreground. As the collector was running 24/7, some applications were open for multiple hours while the computer was on but not being used. So, we decided to filter out and omit every entry longer than three hours. Using this cleaned up data, we found the counts, sum, average, and standard deviation of each application's time in the foreground. We then used the data to create visuals and plots of statistics. In the coming weeks, we will use this data to create our model and predictions.