1. Print everything in working dictionary problem1
2. Cd~

Cd datasets

Ls -l /active data

pwd

1. 1. cd~: home directory
   2. mkdir ./problem\_set\_1: make a directory called problem\_set\_1
   3. touch submission: makes text file called submission
   4. cd..: change current dictionary to parent dictionary
   5. pwd: print working dictionary

cp ~ / config.txt .

cd /

mkdir preferences

cd preferences

cp ~/ prefs.txt

head config.txt

less config.txt

1. mv datafile.csv Documents

cp datafile.csv ~/Documents

rm datafile.csv Documents

1. A picture of the github cat is printed out

From delorean import Delorean

EST= “US/Eastern”

D=Delorean (timezone=EST)

Print(d)

Prints: (2020,2,05,12,32,21,678303),timezone= ‘US/Eastern’

1. Wget <https://finances.worldbank.org/api/views/yu93-ayrw/rows.csv?accessType=DOWNLOAD>
2. Grep Total rows.csv?accessType=DOWNLOAD-c

33

1. priyankapanati-1:Desktop priyankapanati$ cat rows.csv?accessType=DOWNLOAD > distilledExpenditures.csv

1. Download the data in the Data/Financial Data folder on Canvas. Open the file and take a peek at the data dictionary. What do you think this data is used for?

This data is used to measure someone's credit line and their chances to be successful with financial data. To me by measuring the columns among each other, it looks like they are able to create a well rounded prediction as to how the person handles money and the kinds of things that are affecting their finances.

2. Create a function called loadAndCleanData that takes as an argument a filename and returns a Pandas dataframe. That dataframe should contain the data from the CSV file cleaned such that any cells missing data, containing a NaN value or the string "NA" are filled with 0s (this is a technique called zero-filling that we will talk about shortly!)

3. Add a line to your Python file that uses the function to load in the creditData.csv file from Canvas when the Python script is run.

4. Now that you've got your data loading, you can generate probability density functions for each feature. These PDFs will tell you the probability of a given feature occurring based on our data. You can use Kernel Density Estimation (KDE) to do this. Write a function called computePDF that takes as arguments a target feature and a dataset and generates a KDE plot for each feature in your data (hint: check out the plot.kde function here (Links to an external site.)). You will need to import matplotlib.pyplot as plt and use plt.show() to make the graphs appear. Call that feature on each column of your dataset when you run your script.

5. Given the skews that you see in your data, you might want to step back and take a look at what's actually in your data. You can look at the distribution of values in the columns. This will help you understand what data you have. To do this, write a function called viewDistribution that takes in the name of a column and a dataframe and shows a histogram of values in that column (hint: check out the hist function here (Links to an external site.)).  Comment out your computePDF function call and instead use viewDistribution to look at the distribution of each column in your dataset when the Python script is run. This should come after you call the loadAndCleanData function. Notice anything strange about some of these histograms?

6. When your data distributions are radically skewed, you can use a log scale to help reveal data that is otherwise too sparse to see. Write a new version of the viewDistribution function called viewLogDistribution to show the log distribution of each column. Add this function call after your viewDistribution call to view the regular and log distributions of each feature.

7. Use the two distributions to identify three bins per column that divide your data into roughly equal numbers. What are those bins? Note you do not need bins for "SeriousDlqin2yrs" as that is the feature you are modeling (it is your dependent variable).

8. Write a function called computeDefaultRisk that takes four arguments---a column name, a bin (as an array [start,end]), a target feature, and a dataframe---and returns the probability that someone will be at least 90 days delinquent on their account (in other words, "SeriousDlqin2yrs" = 1). Keep in mind that this probability is conditional, that means you'll want to use Bayes rule to compute it. In plain English, you should compute the probability that a loan will become seriously delinquent given your target feature falls into the bin range. For example, if I'm looking at ages between 0 and 40, I want to compute the probability that a loan will go into serious delinquency given the applicant is between 0 and 40.

9. Print out the risk of default for each of the feature bins in your dataset. Note it's helpful to label these with the feature and bins such that you can better understand your output.