Student Name: Yamini Lakshmi Narasimhan

Student Netid: yl9822

Foundations of Data Science Fall 2022 - Homework 0 (30 points)

Part 1: Pre-class survey (5 Points)

- Fill in this survey which will help our course team understand student backgrounds and interests.

Part 2: Case study (5 Points)

- Read this article in the New York Times.
- Use what we've learned in class and from the book to describe how one could set Target's problem up as a predictive modeling problem, such that they could have gotten the results that they did. Formulate your solution as a proposed plan using our data science terminology. Include aspects of the Data Science Workflow that you see as relevant to solving the problem. Be precise but concise.

Place your answer here!

Part 3: Exploring data in the command line (4 Points - 1 Point Each)

- For this part we will be using the data file `"loansData.csv"`. This file

consists of records that pertain to some loan records in a local bank. There are 15 comma separated columns in this order:

> > `CustNUm`, `Amount.Requested`, `Amount.Funded.By.Investors`, `Interest.Rate`, `Loan.Length`, `Loan.Purpose`, `Debt.To.Income.Ratio`, `State`, `Home.Ownership`, `Monthly.Income`, `FICO.Range`, `Open.CREDIT.Lines`, `Revolving.CREDIT.Balance`, `Inquiries.in.the.Last.6.Months`, and `Employment.Length`.

- These fields contain data of type `int`, `float`, and `string`, and you can also locate a file `"data/loansData_columns.csv"` in the data folder containing all the column names for easy reference. Answer the following questions using Linux/Unix bash commands. All questions can be answered in one line (sometimes, with pipes)! Some questions will have many possible solutions. Don't forget that in iPython notebooks you must prefix all bash commands with an exclamation point, i.e. `"!command arguments"`.

> > **1**. *How many records (lines) are in this file?*

In [1]:

```
# Place your code here
import pandas as pd
!tail -n+2 loansData.csv | wc -l
```

```
2500
```

> > **2**. *How many unique `State` (the 8th field) are in this file? (hint: consider the 'cut' command and use pipe operator '|')*

In [2]:

```
# Place your code here
#len(pd.unique(df["State"]))
!tail -n+2 loansData.csv | cut -f 8 -d ,  | sort | uniq -c  | wc -l
```

```
46
```

> **3**. *Rank all domains by the number of `Loan.Purpose` (the 6th field) they requested in descending order. (hint: consider the 'cut', 'uniq' and 'sort' commands and the pipe operator).*

In [3]:

```
# Place your code here
import pandas as pd
!tail -n+2 loansData.csv| cut -f6 -d',' |sort | uniq -c | sort -nr
```

```
1307 "debt_consolidation"
 444 "credit_card"
 201 "other"
 152 "home_improvement"
 101 "major_purchase"
  87 "small_business"
  50 "car"
  39 "wedding"
  30 "medical"
  29 "moving"
  21 "vacation"
  20 "house"
  15 "educational"
   4 "renewable_energy"
```

> **4**. *List all records which have `FICO.Range` (the 11th field) from 815-819. (hint: this can be done using 'grep')*

In [4]:

```
# Place your code here
!cat loansData.csv | grep '815-819'   #see if you can put column 11 anywhere
```

```
"64884",9000,9000,"6.03%","36 months","vacation","5.58%","NJ","MORTGAGE",95
83.33,"815-819",11,675,0,"n/a"
"55501",8000,8000,"6.03%","36 months","debt_consolidation","4.51%","OR","MO
RTGAGE",3500,"815-819",9,6737,0,"10+ years"
"93374",16500,16500,"6.03%","36 months","debt_consolidation","22.65%","CA",
"MORTGAGE",5416.67,"815-819",17,14835,0,"10+ years"
"90568",4800,4800,"6.62%","36 months","car","10.42%","TX","MORTGAGE",7291.6
7,"815-819",14,0,0,"< 1 year"
"80302",16800,16800,"7.90%","60 months","debt_consolidation","3.34%","FL","
MORTGAGE",10666.67,"815-819",7,4757,0,"10+ years"
"5906",12800,12787.71,"8.94%","36 months","debt_consolidation","0.18%","AZ"
,"MORTGAGE",2833.33,"815-819",7,306,0,"4 years"
```

Part 4: Dealing with data Pythonically (16 Points)

In [5]:

```
# You might find these packages useful. You may import any others you want!
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

> **1. (1 Point)** *Load the data set* `"data/ads_dataset.tsv"` *and load it into a Python Pandas data frame called* `ads`.

In [6]:

```
# Place your code here
ads = pd.read_csv("new_ads_dataset.tsv",sep='\t')
ads.head()
print(ads.columns)
print(ads.dtypes)
```

```
Index(['is_video_user', 'video_freq', 'call_freq', 'video_interval',
       'call_interval', 'expected_video_time', 'expected_call_time',
       'last_bill', 'next_bill', 'multiple_video', 'multiple_carrier',
       'uniq_urls', 'num_texts', 'is_churn'],
      dtype='object')
is_video_user            int64
video_freq             float64
call_freq                int64
video_interval           int64
call_interval          float64
expected_video_time      int64
expected_call_time     float64
last_bill                int64
next_bill                int64
multiple_video           int64
multiple_carrier         int64
uniq_urls                int64
num_texts                int64
is_churn                 int64
dtype: object
```

> > **2. (4 Points)** *Write a Python function called* `getDfSummary()` *that does the following:*

- *Takes as input a data frame*
- *For each variable in the data frame calculates the following features:*
  - `number_nan` *to count the number of missing not-a-number values*
  - *Ignoring missing, NA, and Null values:*
    - `number_distinct` *to count the number of distinct values a variable can take on*
    - `mean`, `max`, `min`, `std` *(standard deviation), and* `25%`, `50%`, `75%` *to correspond to the appropriate percentiles*
- *All of these new features should be loaded in a new data frame. Each row of the data frame should be a variable from the input data frame, and the columns should be the new summary features.*
- Returns this new data frame containing all of the summary information

italicized text

> > **Hint:** *The pandas* `describe()` *(manual page) method returns a useful series of values that can be used here.*

In [7]:

```
import time
import copy
def getDfSummary(input_data):
    output_data={"Variable":[],"count":[],"number_nan":[],"number distinct":
    for a in input_data.columns:
        input_data_not_null = input_data.dropna(subset=[a])[a]
        output_data["Variable"].append(a)
        output_data["number distinct"].append(len(input_data_not_null.unique
        output_data["number_nan"].append(len(input_data)-len(input_data_not_
        for x,y in input_data[a].describe().items():
            output_data[x].append(y)

    return pd.DataFrame.from_dict(output_data)

ads_processed = getDfSummary(ads)
ads_processed
```

Out[7]:

	Variable	count	number_nan	number distinct	mean	max	min
0	is_video_user	54584.0	0	2	0.042632	1.00000	0.0000
1	video_freq	2327.0	52257	10	1.240653	15.00000	1.0000
2	call_freq	54584.0	0	64	1.852777	84.00000	0.0000
3	video_interval	54584.0	0	121	2.570533	120.00000	0.0000
4	call_interval	54584.0	0	5886	5.825610	184.91670	0.0000
5	expected_video_time	54584.0	0	134	-0.494174	55.00000	-78.0000
6	expected_call_time	54584.0	0	15135	-10.210786	91.40192	-187.6156
7	last_bill	54584.0	0	189	64.729335	188.00000	0.0000
8	next_bill	54584.0	0	189	64.729335	188.00000	0.0000
9	multiple_video	54584.0	0	2	0.021563	1.00000	0.0000
10	multiple_carrier	54584.0	0	2	0.277444	1.00000	0.0000
11	uniq_urls	54584.0	0	207	86.569343	206.00000	-1.0000
12	num_texts	54584.0	0	4628	720.657592	37091.00000	1.0000
13	is_churn	54584.0	0	2	0.004635	1.00000	0.0000

> > **3. (1 Point)** *How long does it take for your `getDfSummary()` function to work on your `ads` data frame? Show us the results below.*

> > **Hint:** `%timeit getDfSummary(ads)`

In [8]:

```
%timeit getDfSummary(ads)
```

```
26 ms ± 914 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

> > **4. (2 Points)** *Using the results returned from `getDfSummary()`, which fields, if any, contain missing `NaN` values?*

In [9]:
```
print("No values in getDfSummary output contain NaN whereas video_freq colum

ads_null = ads.isnull()
ads_has_NaN = ads_null.any(axis=1)
ads_has_NaN_rows = ads[ads_has_NaN]


ads_has_NaN_processsed = getDfSummary(ads_has_NaN_rows)

display(ads_has_NaN_processsed)
print("\n\n\n")
print("BEFORE")
print(ads["video_freq"].value_counts(dropna=False))
print("\n\n\n")
print("AFTER")
print(ads_has_NaN_rows["video_freq"].value_counts(dropna=False))
```

```
No values in getDfSummary output contain NaN whereas video_freq column cont
ains 52257 NaN(s) out of 54584
```

	Variable	count	number_nan	number distinct	mean	max	min
0	is_video_user	52257.0	0	1	0.000000	0.00000	0.0000
1	video_freq	0.0	52257	0	NaN	NaN	NaN
2	call_freq	52257.0	0	48	1.651549	84.00000	1.0000
3	video_interval	52257.0	0	1	0.000000	0.00000	0.0000
4	call_interval	52257.0	0	5112	5.686388	184.91670	0.0000
5	expected_video_time	52257.0	0	1	0.000000	0.00000	0.0000
6	expected_call_time	52257.0	0	13351	-9.669298	91.40192	-187.6156
7	last_bill	52257.0	0	189	65.741317	188.00000	0.0000
8	next_bill	52257.0	0	189	65.741317	188.00000	0.0000
9	multiple_video	52257.0	0	1	0.000000	0.00000	0.0000
10	multiple_carrier	52257.0	0	2	0.255602	1.00000	0.0000
11	uniq_urls	52257.0	0	207	86.656180	206.00000	-1.0000
12	num_texts	52257.0	0	4570	721.848518	37091.00000	1.0000
13	is_churn	52257.0	0	2	0.003024	1.00000	0.0000

```
BEFORE
NaN     52257
1.0      1980
2.0       244
3.0        55
4.0        20
5.0        17
6.0         5
11.0        2
7.0         2
15.0        1
8.0         1
Name: video_freq, dtype: int64
```

```
AFTER
NaN    52257
Name: video_freq, dtype: int64
```

> > **5. (4 Points)** *For the fields with missing values, does it look like the data is missing at random? Are there any other fields that correlate perfectly, or predict that the data is missing? If missing, what should the data value be?*
> >
> > **Hint:** *create another data frame that has just the records with a missing value. Get a summary of this data frame using `getDfSummary()` and compare the differences. Do some feature distributions change dramatically?*

In [10]:

```
print("Question : For the fields with missing values, does it look like the
print("Answer : Nope not in random")
print("\n\n")
print("Question : Are there any other fields that correlate perfectly, or pr
print("Answer : video_freq is NaN when there is no corresponding is_video_us
Question : For the fields with missing values, does it look like the data i
s missing at random?
Answer : Nope not in random
```

```
Question : Are there any other fields that correlate perfectly, or predict
that the d--ata is missing? If missing, what should the data value be?
Answer : video_freq is NaN when there is no corresponding is_video_user = 0
and video_interval =0 so possible value for video_freq could be any number
that is not possible at all like 0 or any negative number or even text 'Not
Applicable' that shows its not possible to have a video_freq when there is
no video
```

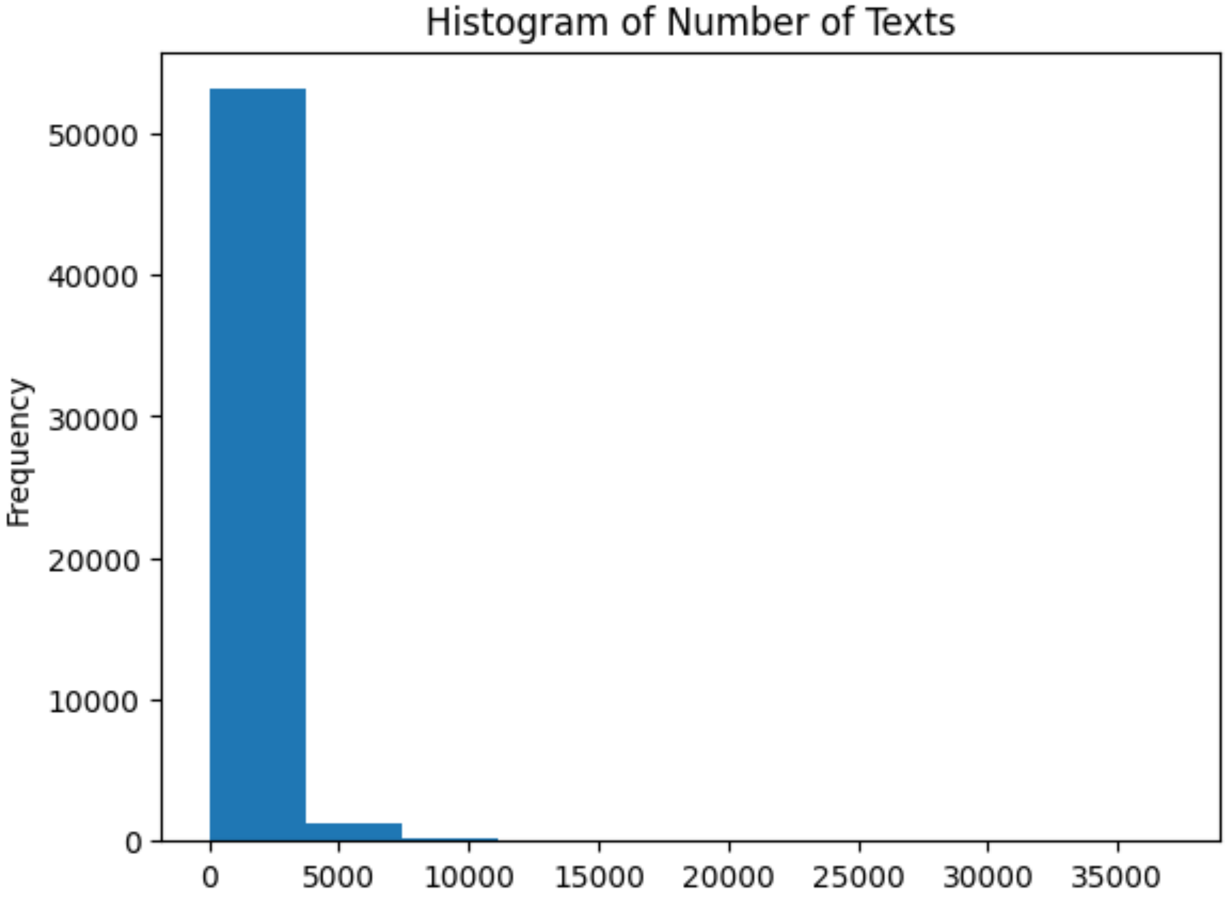
> > **6**. **(2 Points)** *Which variables are binary?*

In [11]:
```
# Place your code here
binary_cols = [col for col in ads.columns if len(ads[col].unique())==2]
print(binary_cols)
```

```
['is_video_user', 'multiple_video', 'multiple_carrier', 'is_churn']
```

> > **7**. **(0.5 Point)** *Let's take a deeper look into one of the features, the `num_texts`, which stands for the number of text messages. Let's try and understand the distribution of this field. We can do this using the hist() method and matplotlib. Draw a histogram graph of `num_texts` from the dataframe `ads`, set the title of the graph as `'Histogram of Number of Texts'`.*

In [12]:
```
# Place your code here
import matplotlib.pyplot as plt
ax = ads["num_texts"].plot(kind='hist')
plt.title('Histogram of Number of Texts')
plt.show()
```



> **8. (1.5 Point)** *How would you characterize the shape of this distribution? Is there anything we can do to the texts variable to make the distribution more bell curved?*

> **Hint:** *Let's create a new column in the dataframe called* `'log_num_texts'` *and print a histogram* `'Histogram of Log(Num Texts)'` *of it. What might be some advantages of making such a transformation?*

In [13]:

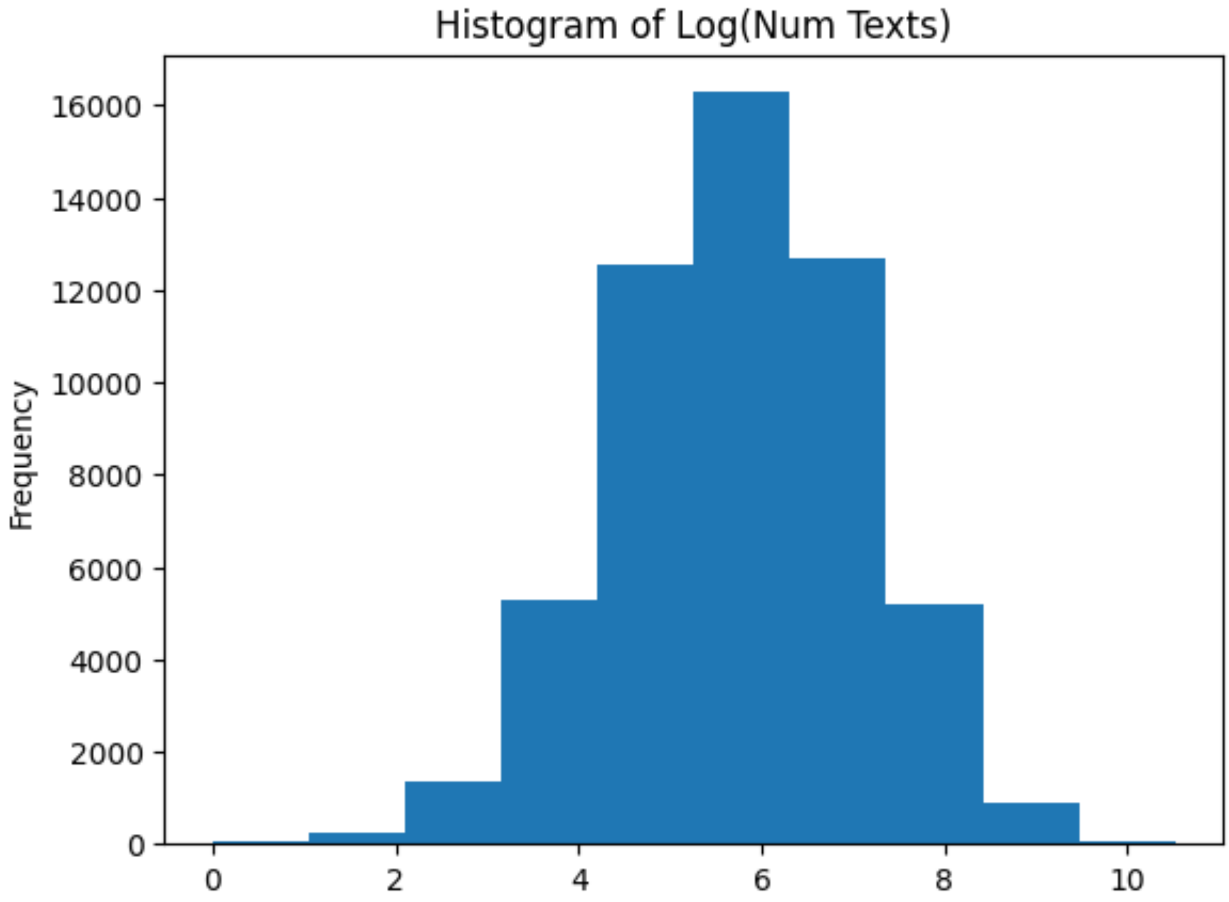
```
# Place your code and response here
import matplotlib.pyplot as plt
import math
import numpy as np

print("How would you characterize the shape of this distribution? Is there a
print("The graph is more left inclined by that I mean majority of values exi
print("\n\n\n")


ads["log_num_texts"] = np.log(ads["num_texts"])
ax = ads["log_num_texts"].plot(kind='hist')
plt.title('Histogram of Log(Num Texts)')
plt.show()
print("\n\n\n")


import statistics
print("Mean ", ads["log_num_texts"].describe()["mean"])
print("Mode ", statistics.mode(ads["log_num_texts"]))
print("Median ",statistics.median(ads["log_num_texts"]))
print("Was just checking if its normal distribution")
```

```
How would you characterize the shape of this distribution? Is there anythin
g we can do to the texts variable to make the distribution more bell curved
?
The graph is more left inclined by that I mean majority of values exist bet
ween 0-5k than from 5k-38k
```



```
Mean  5.741782004917109
Mode  4.248495242049359
Median  5.765191102784844
Was just checking if its normal distribution
```

End of Homework 0

In [ ]: