DataFest Dataset

Nicolas Huang

4/14/23

Capital University : R-μ-Ready

In [39]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
```

1. Questions

In [27]:
```
# Import Questions
questionRoot = pd.read_csv("./data/questions.csv", on_bad_lines='skip', low_memory=False)
questions = questionRoot
```

In [28]:
```
questions.head(10)
```

...

In [29]:
```
# Get a count of how many questions are in question
totalQuestions = questions['Id'].count()
totalQuestions
```

...

1. Converting to DTF DataType is Resource Intensive! (Run Once)

In [30]:
```
# Convert Times to DTF object
questions['AskedOnUtc'] = pd.to_datetime(questions['AskedOnUtc'])
questions['TakenOnUtc'] = pd.to_datetime(questions['TakenOnUtc'])
questions['ClosedOnUtc'] = pd.to_datetime(questions['ClosedOnUtc'])
```

Free Legal has a Automatic 30 Day Close Policy, Set Closed On to 30 Days after Open

In [31]:
```
# If Closed On is Null, and isn't taken by any lawyer, set closed on to 30 days after asked on
```

In [32]:
```
# Start time
questionTimeStart = questions['AskedOnUtc']
# questionTimeStart.head()

# End time
questionTimeEnd = questions['ClosedOnUtc']
# questionTimeStart.head()

# Taken On Time
questionTimeTaken = questions['TakenOnUtc']
# questionTimeTaken.head()
# pd.isnull(questions['TakenOnUtc'][0])  # Check if the question was never taken on for the first row
```

1.1 TimeDuration of Questions & How Many left unanswered

In [33]:
```
# Question Duration
questions['TimeOpen'] = questionTimeEnd - questionTimeStart

# questions['TimeOpen'].head()
questionTimeDuration = questions['TimeOpen']

# questionTimeDuration.head()
print(f"Statistical Summary of Question Duration (aka how long to remediate)")
questionTimeDuration.describe()
```

```
Statistical Summary of Question Duration (aka how long to remediate)
```

Out[33]:
```
count                        199085
mean     26 days 05:30:32.554737926
std      44 days 20:10:07.767320240
min                 0 days 00:00:08
25%                 5 days 01:47:32
50%                12 days 14:18:37
75%                33 days 19:49:03
max              2530 days 20:41:03
Name: TimeOpen, dtype: object
```

In [34]:
```
# Count how many questions weren't taken on by lawyers

# Filter only the questions that weren't taken on by a lawyer
notTakenByLawyer = questions.loc[lambda df: pd.isnull(questions['TakenOnUtc']) == True]
notTakenByLawyer.head(10)

# Select only the filtered questions that were never closed by an attorney
notTakenByLawyer = notTakenByLawyer['ClosedByAttorneyUno']
unansweredTotal = notTakenByLawyer.count()
hitMissRatio = (unansweredTotal / totalQuestions) * 100
print(f"Total Number of Questions that weren't taken by a lawyer \nand went stale is: {hitMissRatio:.2f}%, or
```

```
Total Number of Questions that weren't taken by a lawyer
and went stale is: 2.58%, or 5240 cases
```

In [35]:
```
# Check if any cases are left open
questionsLeftOpen = questions.loc[lambda df: pd.isnull(questions['ClosedOnUtc'])]

print(f" Number of Cases left open: {questionsLeftOpen['ClosedOnUtc'].isna().count()}")
```

```
 Number of Cases left open: 3794
```

Group Hit or Miss, Question Time Duration, Number of Cases Open, and Statistical Summary by State

In [36]:
```
# Save a smaller df
groupStateRoot = questions[['Id', 'StateAbbr', 'AskedOnUtc', 'ClosedOnUtc', 'TakenOnUtc', 'Category', 'TimeOpe
                            'ClosedByAttorneyUno']]
# groupStateRoot.head(100)

# Select only cases that aren't stale (aka, never closed)
# groupState = groupStateRoot.loc[lambda df: pd.isnull(groupStateRoot['ClosedOnUtc']) == False]
groupState = groupStateRoot

# Create summary of each state
timeDurationByState = groupState.groupby('StateAbbr')['TimeOpen'].describe()
# timeDurationByState.head(10)

# Create total cases per state
stateTotalCases = groupState.groupby('StateAbbr')['Id'].count()

# Create total cases not taken per state
stateTotalUnanswered = groupState.groupby('StateAbbr')['TakenOnUtc'].count()  # This holds number of cases tak
# Take the difference of taken on and total cases to get total cases not taken
stateTotalUnanswered = stateTotalCases - stateTotalUnanswered

# Add columns to our dataframe
timeDurationByState['TotalUnanswered'] = stateTotalUnanswered
timeDurationByState['HitMissRatio'] = (stateTotalUnanswered / stateTotalCases)
timeDurationByState.head(10)
```

...

In [37]:
```
# Export the TimeDurationbyState Dataframe
timeDurationByState.to_csv("QuestionAnaylticsByState.csv", sep=",")
```

2. Attorney Time Entries

In [38]:
```
# Import Attorneytimeentries
attorneyLogonRoot = pd.read_csv("./data/attorneytimeentries.csv", on_bad_lines ='skip', low_memory=False)
attorneyLogonRoot.head()
```

...

In [32]:
```
# Convert Entered On to DTF Format
attorneyLogonRoot['EnteredOnUtc'] = pd.to_datetime(attorneyLogonRoot['EnteredOnUtc'])
```

In [33]:
```
# Keep only Hour and Minute, save it to Logon (Minutes are in decimal format!)
attorneyLogonRoot['Hour'] = attorneyLogonRoot['EnteredOnUtc'].dt.hour
attorneyLogonRoot['Minute'] = attorneyLogonRoot['EnteredOnUtc'].dt.minute
attorneyLogonRoot['LogonTime'] = attorneyLogonRoot['Hour'] + (attorneyLogonRoot['Minute'] / 60)
attorneyLogonRoot.head(10)
```

...

In [34]:
```
# Find Analytical Summary of Logon Times per State
attorneyLogonByState = attorneyLogonRoot.groupby('StateAbbr')['LogonTime'].describe(datetime_is_numeric = True
attorneyLogonByState.head(40)
```

Out[34]:

StateAbbr	count	mean	std	min	25%	50%	75%	max
AK	573.0	13.347673	8.241854	0.033333	4.616667	17.183333	20.000000	23.983333
AL	568.0	16.333421	6.146598	0.366667	15.783333	17.766667	20.341667	23.916667
AR	2008.0	16.426569	6.211651	0.000000	15.345833	18.066667	20.583333	23.983333
AZ	659.0	16.875089	6.009446	0.016667	15.058333	18.533333	20.983333	23.950000
CA	1007.0	14.179345	8.251565	0.033333	5.350000	18.066667	21.141667	23.983333
CT	1080.0	14.257762	6.591995	0.016667	9.500000	15.933333	19.237500	23.933333
FL	13345.0	16.817106	4.433074	0.000000	14.766667	17.316667	19.883333	23.983333
GA	1273.0	15.234158	6.286257	0.050000	13.200000	16.733333	20.100000	23.966667
HI	2505.0	13.399208	8.581020	0.000000	4.000000	17.900000	21.150000	23.983333
IA	701.0	15.605706	6.579780	0.016667	14.783333	17.533333	20.033333	23.966667
IL	8783.0	15.830259	6.303317	0.000000	14.700000	17.500000	20.033333	23.983333
IN	5498.0	15.653707	5.198377	0.000000	13.704167	16.483333	19.233333	23.983333
KS	18.0	16.974074	6.873762	0.016667	14.950000	18.683333	21.158333	23.950000
LA	3134.0	15.673011	6.733690	0.000000	14.816667	17.466667	20.312500	23.983333
MA	3794.0	15.428075	6.127187	0.000000	14.183333	16.916667	19.612500	23.983333
MD	1487.0	15.809740	6.140789	0.000000	14.375000	16.950000	20.133333	23.983333
ME	2275.0	16.259495	5.170523	0.000000	14.383333	17.133333	19.766667	23.983333
MI	50.0	17.105667	4.348615	6.133333	15.979167	18.466667	19.758333	22.550000
MO	7779.0	14.565784	7.103068	0.000000	11.416667	16.716667	19.983333	23.983333
MS	351.0	15.720513	5.806241	0.183333	14.550000	17.516667	19.091667	23.983333
NC	5884.0	15.058492	7.237368	0.000000	13.350000	17.191667	20.633333	23.983333
NE	1409.0	16.539614	5.988372	0.000000	15.000000	18.050000	20.600000	23.983333
NH	1103.0	16.663735	4.340399	0.016667	14.233333	17.083333	19.975000	23.883333
NJ	1022.0	13.056800	7.755788	0.016667	3.350000	15.333333	19.329167	23.966667
NM	1544.0	16.997172	5.466079	0.016667	15.204167	18.091667	20.720833	23.950000
NY	5553.0	15.377129	5.974196	0.000000	13.133333	16.583333	19.600000	23.983333
OK	1949.0	15.291030	6.727572	0.016667	14.150000	17.183333	19.816667	23.950000
PA	157.0	16.172718	5.790836	0.016667	14.466667	16.950000	20.083333	23.666667
SC	4693.0	16.500721	4.589176	0.000000	14.050000	16.983333	20.066667	23.983333
SD	1767.0	12.359423	7.766261	0.000000	3.833333	15.166667	18.408333	23.983333
TN	7913.0	15.345358	6.347614	0.000000	13.433333	16.900000	19.983333	23.983333
TX	9653.0	14.615424	6.776870	0.000000	12.966667	15.983333	19.750000	23.983333
US	117.0	15.683191	5.841043	0.283333	13.733333	17.316667	20.066667	23.883333
UT	518.0	15.640122	6.378380	0.000000	14.337500	17.041667	20.100000	23.916667
VA	4633.0	15.495859	6.315705	0.016667	13.966667	16.933333	19.983333	23.966667
VT	690.0	14.434106	6.377464	0.050000	12.016667	15.950000	19.162500	23.900000
WI	6236.0	15.862358	6.417628	0.000000	14.466667	17.416667	20.350000	23.983333
WV	2169.0	15.907162	5.122256	0.083333	14.250000	16.466667	19.116667	23.950000
WY	715.0	15.680466	7.029822	0.000000	13.858333	17.816667	20.875000	23.966667

In [35]:
```
# Find the average hours of work done by attorneys for each state
hoursByState = attorneyLogonRoot.groupby('StateAbbr')['Hours'].mean()
```

In [36]:
```
# Add hoursByState to attorneyLogonByState
attorneyLogonByState['AverageHours'] = hoursByState
attorneyLogonByState.head(10)
```

Out[36]:

StateAbbr	count	mean	std	min	25%	50%	75%	max	AverageHours
AK	573.0	13.347673	8.241854	0.033333	4.616667	17.183333	20.000000	23.983333	0.322164
AL	568.0	16.333421	6.146598	0.366667	15.783333	17.766667	20.341667	23.916667	0.287148
AR	2008.0	16.426569	6.211651	0.000000	15.345833	18.066667	20.583333	23.983333	0.264193
AZ	659.0	16.875089	6.009446	0.016667	15.058333	18.533333	20.983333	23.950000	0.402731
CA	1007.0	14.179345	8.251565	0.033333	5.350000	18.066667	21.141667	23.983333	1.301092
CT	1080.0	14.257762	6.591995	0.016667	9.500000	15.933333	19.237500	23.933333	0.373056
FL	13345.0	16.817106	4.433074	0.000000	14.766667	17.316667	19.883333	23.983333	0.276496
GA	1273.0	15.234158	6.286257	0.050000	13.200000	16.733333	20.100000	23.966667	0.390888
HI	2505.0	13.399208	8.581020	0.000000	4.000000	17.900000	21.150000	23.983333	0.354052
IA	701.0	15.605706	6.579780	0.016667	14.783333	17.533333	20.033333	23.966667	0.239230

In [37]:
```
# Import Attorney's For Attorney Count by state
attorneys = pd.read_csv("./data/attorneys.csv", on_bad_lines='skip', low_memory=False)
```

In [38]:
```
# Count how many attorneys per state
attorneyByState = attorneys.groupby('StateAbbr')['AttorneyUno'].count()
attorneyByState.head(10)
```

Out[38]:
```
StateAbbr
AK     110
AL     138
AR     192
AZ     104
CA     245
CT     103
FL    1103
GA     308
HI     125
IA      67
Name: AttorneyUno, dtype: int64
```

In [39]:
```
# Add Number of Attorneys Per State in the LogonByState
attorneyLogonByState['NumAttorneys'] = attorneyByState
attorneyLogonByState.head(10)
```

Out[39]:

StateAbbr	count	mean	std	min	25%	50%	75%	max	AverageHours	NumAttorneys
AK	573.0	13.347673	8.241854	0.033333	4.616667	17.183333	20.000000	23.983333	0.322164	110
AL	568.0	16.333421	6.146598	0.366667	15.783333	17.766667	20.341667	23.916667	0.287148	138
AR	2008.0	16.426569	6.211651	0.000000	15.345833	18.066667	20.583333	23.983333	0.264193	192
AZ	659.0	16.875089	6.009446	0.016667	15.058333	18.533333	20.983333	23.950000	0.402731	104
CA	1007.0	14.179345	8.251565	0.033333	5.350000	18.066667	21.141667	23.983333	1.301092	245
CT	1080.0	14.257762	6.591995	0.016667	9.500000	15.933333	19.237500	23.933333	0.373056	103
FL	13345.0	16.817106	4.433074	0.000000	14.766667	17.316667	19.883333	23.983333	0.276496	1103
GA	1273.0	15.234158	6.286257	0.050000	13.200000	16.733333	20.100000	23.966667	0.390888	308
HI	2505.0	13.399208	8.581020	0.000000	4.000000	17.900000	21.150000	23.983333	0.354052	125
IA	701.0	15.605706	6.579780	0.016667	14.783333	17.533333	20.033333	23.966667	0.239230	67

In [40]:
```
attorneyLogonRoot.head()
```

. . .

In [41]:
```
# Import Attorneys
attorneyRoot = pd.read_csv("./data/attorneys.csv", on_bad_lines='skip', low_memory=False)
attorneyRoot.head()
```

...

In [49]:
```
# Check if a unique attorney has or hasn't logged a session
attorneyDidWork = attorneyRoot[['StateAbbr', 'AttorneyUno']]
attorneyDidWork['HasDoneASession'] = attorneyRoot.AttorneyUno.isin(attorneyLogonRoot.AttorneyUno).astype(int)

# Calculate percentage of attorneys having at least ONE session per state
attorneyDidWork = attorneyDidWork.groupby('StateAbbr')['HasDoneASession'].describe()

# attorneyDidWork.head(10)

# Save only necessary columns
attorneyDidWorkFinal = attorneyDidWork
attorneyDidWorkFinal['NumberOfAttorneys'] = attorneyDidWork['count']
attorneyDidWorkFinal['PercentageAttorneyDidASession'] = attorneyDidWork['mean']
attorneyDidWorkFinal = attorneyDidWork[['NumberOfAttorneys', 'PercentageAttorneyDidASession', 'std']]

attorneyDidWorkFinal.head(10)
```

...

In [50]:
```
# Export df to csv for visualization
attorneyLogonByState.to_csv("attorneyAnalyticsPerState.csv", sep=",")
attorneyDidWorkFinal.to_csv("PercentageOfAttorneysDoingASessionPerState.csv", sep=",")
```

X. Questions Post (40,000 Rows Bad, Clean the text)

In [ ]:
```
questionPostRoot = pd.read_csv("./data/questionposts.csv", on_bad_lines='skip', low_memory=False)
```

In [ ]:
```
questionPostRoot.head(10)
```

In [ ]:
```
questionPostRoot['PostText'][14255]
```

In [ ]:
```
questionPost = questionPostRoot[['Id', 'StateAbbr', 'QuestionUno', 'CreatedUtc']]
```

In [ ]:
```
# questionPostRoot["CreatedUtc"][73]
questionPostRoot['PostText'] = questionPostRoot['PostText'].str.replace("\'", '')
questionPostRoot['PostText'] = questionPostRoot['PostText'].str.replace("\"", '')
questionPostRoot['PostText'] = questionPostRoot['PostText'].str.replace(",", '')
questionPostRoot['PostText'] = questionPostRoot['PostText'].str.replace("/", '')
questionPostRoot['PostText'] = questionPostRoot['PostText'].str.replace("\\", '')
questionPostRoot['PostText'] = questionPostRoot['PostText'].str.replace("#", '')
questionPostRoot['PostText'] = questionPostRoot['PostText'].str.replace("$", '')
questionPostRoot['PostText'][14255]
questionPost.to_csv("questionpostTabbed.csv", sep=",")
```