Government Supplier Prediction Project

Alana Rutherford, 500953715, Supervisor Derya Kici, Ph.D, November 28, 2021

Data Import

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import zipfile

with zipfile.ZipFile('data/tpsgc-pwgsc_co-ch_tous-all.zip', 'r') as zip_ref:
    zip_ref.extractall('data')
contractHistory = pd.read_csv('data/tpsgc-pwgsc_co-ch_tous-all.csv', usecols = range(0,42))
```

General Data Description

In [ ]:
```
contractHistory.columns
```

Out[ ]:
```
Index(['contract-number', 'amendment-number', 'award-date', 'expiry-date',
       'contract-value', 'gsin', 'gsin-description_en', 'gsin-description_fr',
       'competitive-tender_en', 'competitive-tender_fr',
       'limited-tender-reason', 'limited-tender-reason-description_en',
       'limited-tender-reason-description_fr', 'solicitation-procedure',
       'solicitation-procedure-description_en',
       'solicitation-procedure-description_fr', 'trade-agreement',
       'trade-agreement-description_en', 'trade-agreement-description_fr',
       'supplier-standardized-name', 'supplier-operating-name',
       'supplier-legal-name', 'supplier-address-city',
       'supplier-address-prov-state', 'supplier-address-postal-code',
       'supplier-address-country', 'organization-employee-count_en',
       'organization-employee-count_fr', 'total-contract-value',
       'number-records', 'end-user-entity_en', 'end-user-entity_fr',
       'contracting-entity-office-name_en',
       'contracting-entity-office-name_fr', 'contracting-address-street-1',
       'contracting-address-street-2', 'contracting-address-city',
       'contracting-address-prov-state', 'contracting-address-postal-code',
       'contracting-address-country', 'procurement-entity-name_en',
       'procurement-entity-name_fr'],
      dtype='object')
```

In [ ]:
```
contractHistory.head()
```

Out[ ]:

	contract-number	amendment-number	award-date	expiry-date	contract-value	gsin	gsin-description_en	gsin-description_fr	competitive-tender_en	competitive-tender_fr	...	contracting-entity-office-name_en	contracting-entity-office-name_fr	contractin addres street
0	F2599-090025/001/MD	000	2009-06-16	2009-10-20	30174.0	N1990	Vessels, Miscellaneous	Bateaux divers	Yes	Oui	...	CCG CENTRAL&ARCTIC RGN	N/D	MARI ENGINEERI
1	F2599-090025/001/MD	001	2009-07-15	2009-10-20	4876.0	N1990	Vessels, Miscellaneous	Bateaux divers	Yes	Oui	...	CCG CENTRAL&ARCTIC RGN	N/D	MARI ENGINEERI
2	F2599-090025/001/MD	002	2009-07-20	2009-10-20	26914.0	N1990	Vessels, Miscellaneous	Bateaux divers	Yes	Oui	...	CCG CENTRAL&ARCTIC RGN	N/D	MARI ENGINEERI
3	E60LP-090002/392/LP	000	2010-01-01	2010-12-31	25000.0	V502B	Hotels, Motels and Commercial Accommodation	Hôtels, motels et logements commerciaux	Yes	Oui	...	CONSOLIDATED PROC LP ICPSS	N/D	PORTAGE 7
4	E60LP-110001/050/LP	000	2012-01-01	2012-12-31	25000.0	V502B	Hotels, Motels and Commercial Accommodation	Hôtels, motels et logements commerciaux	Yes	Oui	...	CONSOLIDATED PROC LP ICPSS	N/D	PORTAGE 7

5 rows × 42 columns

In [ ]:
```
contractHistory.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 441210 entries, 0 to 441209
Data columns (total 42 columns):
 #   Column                                Non-Null Count   Dtype
---  ------                                --------------   -----
 0   contract-number                       441210 non-null  object
 1   amendment-number                      441210 non-null  object
 2   award-date                            441210 non-null  object
 3   expiry-date                           441210 non-null  object
 4   contract-value                        441210 non-null  float64
 5   gsin                                  441199 non-null  object
 6   gsin-description_en                   441199 non-null  object
 7   gsin-description_fr                   441210 non-null  object
 8   competitive-tender_en                 441210 non-null  object
 9   competitive-tender_fr                 441210 non-null  object
 10  limited-tender-reason                 94299 non-null   float64
 11  limited-tender-reason-description_en  94299 non-null   object
 12  limited-tender-reason-description_fr  441210 non-null  object
 13  solicitation-procedure                441210 non-null  object
 14  solicitation-procedure-description_en 441210 non-null  object
 15  solicitation-procedure-description_fr 441210 non-null  object
 16  trade-agreement                       440929 non-null  object
 17  trade-agreement-description_en        436587 non-null  object
 18  trade-agreement-description_fr        441210 non-null  object
 19  supplier-standardized-name            441210 non-null  object
 20  supplier-operating-name               438586 non-null  object
 21  supplier-legal-name                   439980 non-null  object
 22  supplier-address-city                 441091 non-null  object
 23  supplier-address-prov-state           430737 non-null  object
 24  supplier-address-postal-code          438200 non-null  object
 25  supplier-address-country              441210 non-null  object
 26  organization-employee-count_en        441208 non-null  object
 27  organization-employee-count_fr        441210 non-null  object
 28  total-contract-value                  441210 non-null  float64
 29  number-records                        441210 non-null  int64
 30  end-user-entity_en                    441210 non-null  object
 31  end-user-entity_fr                    441210 non-null  object
 32  contracting-entity-office-name_en     431285 non-null  object
 33  contracting-entity-office-name_fr     441210 non-null  object
 34  contracting-address-street-1          441182 non-null  object
 35  contracting-address-street-2          308117 non-null  object
 36  contracting-address-city              441182 non-null  object
 37  contracting-address-prov-state        440905 non-null  object
 38  contracting-address-postal-code       441180 non-null  object
 39  contracting-address-country           441182 non-null  object
 40  procurement-entity-name_en            441210 non-null  object
 41  procurement-entity-name_fr            441210 non-null  object
dtypes: float64(3), int64(1), object(38)
memory usage: 141.4+ MB
```

Attribute Data Description

Includes use of describe to obtain the count and unique, and use of value count to identify the top most frequent attributes. This was completed for all categorical attributes that were not considered duplicates on initial analysis. For the sake of brevity, only the describe function is provided below after the first attribute.

In [ ]:
```
contractHistory.describe()
```

Out[ ]:

	contract-value	limited-tender-reason	total-contract-value	number-records
count	4.412100e+05	94299.000000	4.412100e+05	441210.000000
mean	4.398310e+05	75.398689	8.457397e+06	8.202300
std	1.609191e+07	7.860150	1.061860e+08	18.275149
min	-1.540262e+08	8.000000	-1.985487e+08	1.000000
25%	0.000000e+00	71.000000	2.096325e+04	1.000000
50%	8.715000e+03	71.000000	1.268150e+05	3.000000
75%	6.941075e+04	85.000000	7.749660e+05	6.000000
max	5.221000e+09	90.000000	5.761974e+09	206.000000

In [ ]:
```
contractHistory['contract-number'].describe()
```

Out[ ]:
```
count                  441210
unique                 199675
top       E0208-150548/001/PWZ
freq                      206
Name: contract-number, dtype: object
```

In [ ]:
```
contractHistory['contract-number'].value_counts()
```

Out[ ]:
```
E0208-150548/001/PWZ    206
EN578-110558/001/XL     162
EP008-112560/001/GC     152
```

```
EP008-112560/004/GC     139
EW038-140681/001/PWU    139
                       ...
W0113-10A128/001/BOR      1
5K003-156233/001/WPG      1
5K003-148856/001/WPG      1
W8482-128975/001/GRK      1
U4030-221421/001/HN       1
Name: contract-number, Length: 199675, dtype: int64
```

In [ ]:
```
contractHistory['amendment-number'].describe()
```

Out[ ]:
```
count     441210
unique       237
top          000
freq      174270
Name: amendment-number, dtype: object
```

In [ ]:
```
contractHistory['award-date'].describe()
```

Out[ ]:
```
count         441210
unique          4085
top       2011-01-01
freq            2277
Name: award-date, dtype: object
```

In [ ]:
```
contractHistory['expiry-date'].describe()
```

Out[ ]:
```
count         441210
unique          6859
top       2011-03-31
freq            9500
Name: expiry-date, dtype: object
```

In [ ]:
```
contractHistory['gsin'].describe()
```

Out[ ]:
```
count     441199
unique      5203
top         N7030
freq       26529
Name: gsin, dtype: object
```

In [ ]:
```
contractHistory['gsin-description_en'].describe()
```

Out[ ]:
```
count           441199
unique            5188
top       ADP Software
freq             26529
Name: gsin-description_en, dtype: object
```

In [ ]:
```
contractHistory['competitive-tender_en'].describe()
```

Out[ ]:
```
count     441210
unique         2
top          Yes
freq      346895
Name: competitive-tender_en, dtype: object
```

In [ ]:
```
contractHistory['limited-tender-reason-description_en'].describe()
```

Out[ ]:
```
count                94299
unique                  16
top       Exclusive Rights
freq                  57400
Name: limited-tender-reason-description_en, dtype: object
```

In [ ]:
```
contractHistory['solicitation-procedure-description_en'].describe()
```

Out[ ]:
```
count           441210
unique               4
top       Open Bidding
freq            269648
Name: solicitation-procedure-description_en, dtype: object
```

In [ ]:
```
contractHistory['trade-agreement'].describe()
```

Out[ ]:
```
count     440929
unique        35
top            I
freq      115604
Name: trade-agreement, dtype: object
```

In [ ]:
```
contractHistory['supplier-standardized-name'].describe()
```

Out[ ]:
```
count                                   441210
unique                                   36355
```

```
top       SIMEX DEFENCE INC / DEFENSE SIMEX INC
freq                                       4305
Name: supplier-standardized-name, dtype: object
```

In [ ]:
```
contractHistory['supplier-address-city'].describe()
```

Out[ ]:
```
count     441091
unique      7261
top       Ottawa
freq       80001
Name: supplier-address-city, dtype: object
```

In [ ]:
```
contractHistory['supplier-address-prov-state'].describe()
```

Out[ ]:
```
count      430737
unique         64
top       Ontario
freq       190079
Name: supplier-address-prov-state, dtype: object
```

In [ ]:
```
contractHistory['supplier-address-country'].describe()
```

Out[ ]:
```
count     441210
unique       115
top       Canada
freq      398099
Name: supplier-address-country, dtype: object
```

In [ ]:
```
contractHistory['supplier-address-postal-code'].describe()
```

Out[ ]:
```
count     438200
unique     27226
top       H9R1A6
freq        4305
Name: supplier-address-postal-code, dtype: object
```

In [ ]:
```
contractHistory['organization-employee-count_en'].describe()
```

Out[ ]:
```
count                441208
unique                   14
top       20 to 49 employees
freq                  71520
Name: organization-employee-count_en, dtype: object
```

In [ ]:
```
contractHistory['end-user-entity_en'].describe()
```

Out[ ]:
```
count                                        441210
unique                                          132
top       Public Works and Government Services Canada
freq                                         182963
Name: end-user-entity_en, dtype: object
```

In [ ]:
```
contractHistory['contracting-entity-office-name_en'].describe()
```

Out[ ]:
```
count     431285
unique      3422
top         NDHQ
freq       19931
Name: contracting-entity-office-name_en, dtype: object
```

In [ ]:
```
contractHistory['contracting-address-street-1'].describe()
```

Out[ ]:
```
count                441182
unique                 3140
top       101 COLONEL BY DR.
freq                  25490
Name: contracting-address-street-1, dtype: object
```

In [ ]:
```
contractHistory['contracting-address-street-2'].describe()
```

Out[ ]:
```
count            308117
unique             1561
top       11 LAURIER ST
freq              82908
Name: contracting-address-street-2, dtype: object
```

In [ ]:
```
contractHistory['contracting-address-city'].describe()
```

Out[ ]:
```
count     441182
unique       440
top       OTTAWA
freq      130007
Name: contracting-address-city, dtype: object
```

In [ ]:
```
contractHistory['contracting-address-prov-state'].describe()
```

Out[ ]:
```
count      440905
unique         13
top       Ontario
freq       166823
Name: contracting-address-prov-state, dtype: object
```

In [ ]:
```
contractHistory['contracting-address-postal-code'].describe()
```

Out[ ]:
```
count     441180
unique      1196
top       K1A0S5
freq      100214
Name: contracting-address-postal-code, dtype: object
```

In [ ]:
```
contractHistory['contracting-address-country' ].describe()
```

Out[ ]:
```
count     441182
unique         4
top       Canada
freq      440905
Name: contracting-address-country, dtype: object
```

In [ ]:
```
contractHistory['procurement-entity-name_en'].describe()
```

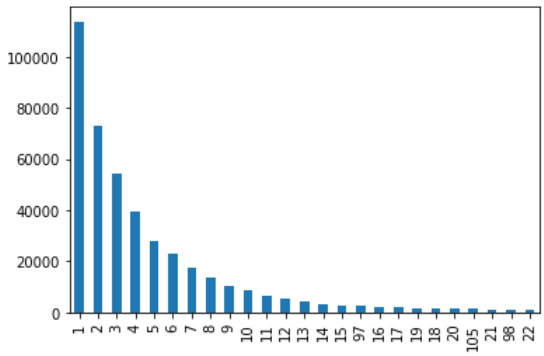
Out[ ]:
```
count                                        441210
unique                                            1
top       Public Works and Government Services Canada
freq                                         441210
Name: procurement-entity-name_en, dtype: object
```

Visualizing Categorical Attributes

A selection of the categorical attributes have been visualized below for the sake of brevity. The full set were initially reviewed.

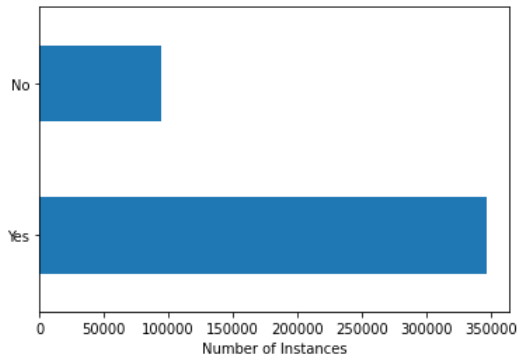
In [ ]:
```
#contractHistory['amendment-number'].value_counts().plot(kind = 'bar')
recordNo = contractHistory['number-records'].value_counts()
recordNo = recordNo[recordNo > 1000]
recordNo.plot(kind = 'bar')
```

Out[ ]:
```
<AxesSubplot:>
```



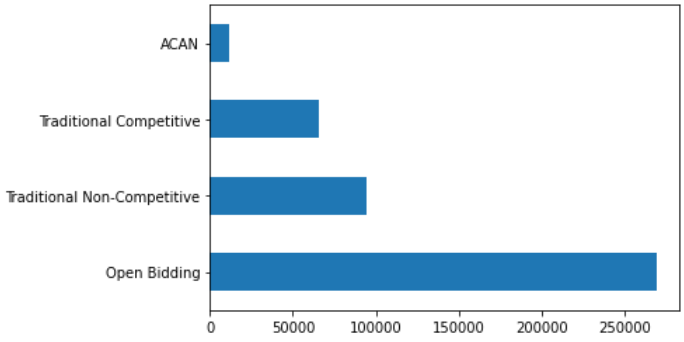
In [ ]:
```
contractHistory['competitive-tender_en'].value_counts().plot(kind = 'barh')
plt.xlabel('Number of Instances')
```

Out[ ]:
```
Text(0.5, 0, 'Number of Instances')
```



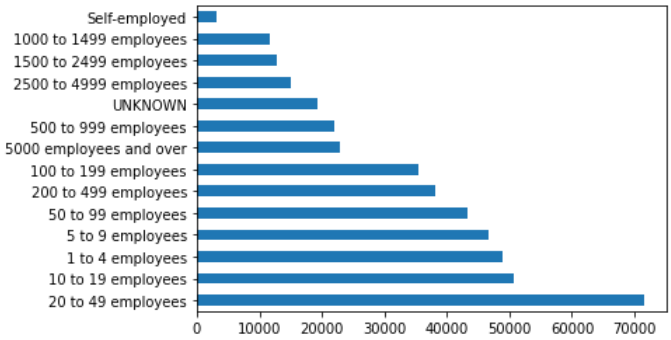
In [ ]:
```
contractHistory['solicitation-procedure-description_en'].value_counts().plot(kind = 'barh')
```

Out[ ]:
```
<AxesSubplot:>
```



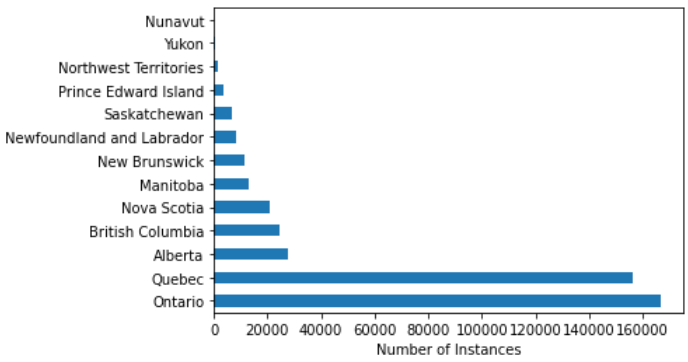
In [ ]:
```
contractHistory['organization-employee-count_en'].value_counts().plot(kind = 'barh')
```

Out[ ]:
```
<AxesSubplot:>
```



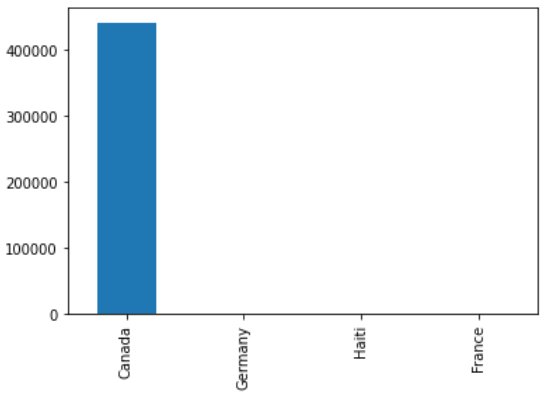
In [ ]:
```
contractHistory['contracting-address-prov-state'].value_counts().plot(kind = 'barh')
plt.xlabel('Number of Instances')
```

Out[ ]:
```
Text(0.5, 0, 'Number of Instances')
```



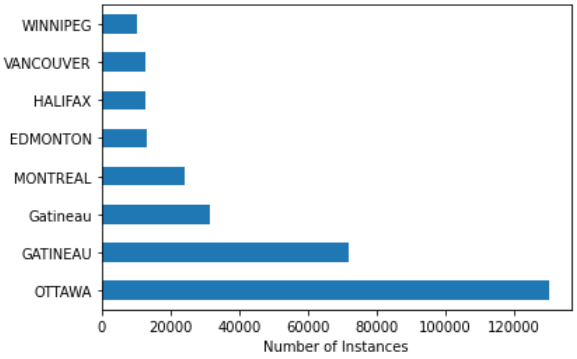
In [ ]:
```
contractHistory['contracting-address-country'].value_counts().plot(kind = 'bar')
```

Out[ ]:
```
<AxesSubplot:>
```



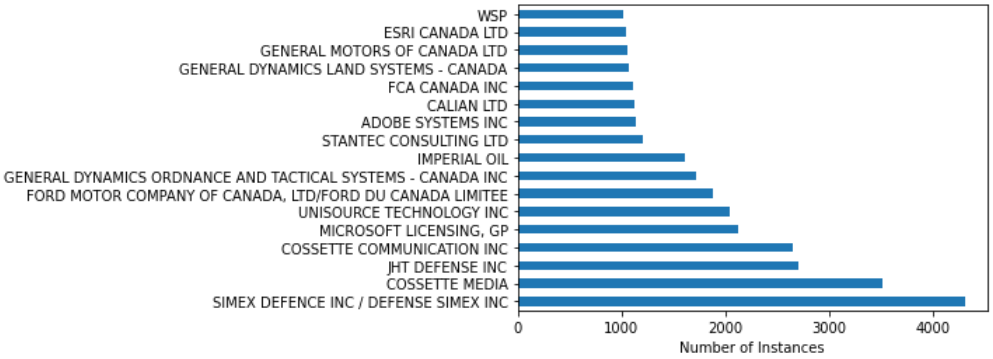
In [ ]:
```
locate = contractHistory['contracting-address-city'].value_counts()
locate = locate[locate > 10000]
locate.plot(kind = 'barh')
plt.xlabel('Number of Instances')
```

Out[ ]:
```
Text(0.5, 0, 'Number of Instances')
```



In [ ]:
```
recordNo = contractHistory['supplier-standardized-name'].value_counts()
recordNo = recordNo[recordNo > 1000]
recordNo.plot(kind = 'barh')
plt.xlabel('Number of Instances')
```

Out[ ]:
```
Text(0.5, 0, 'Number of Instances')
```

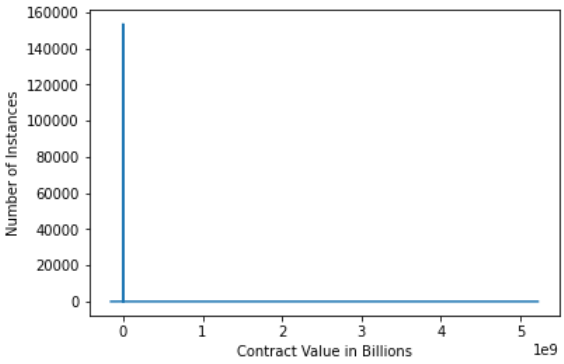


Visualizing Numerical Attributes

In [ ]:
```
contractValue = contractHistory['contract-value'].value_counts().rename_axis('unique_values').reset_index(name='counts')
contractValue = pd.DataFrame(contractValue)
contractValue = contractValue.sort_values(by='unique_values')
```

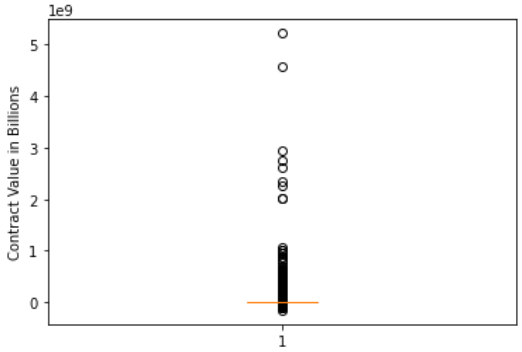
In [ ]:
```
plt.plot(contractValue['unique_values'],contractValue['counts'])
plt.ylabel('Number of Instances')
plt.xlabel('Contract Value in Billions')
```

Out[ ]:
```
Text(0.5, 0, 'Contract Value in Billions')
```



In [ ]:
```
plt.boxplot(contractValue['unique_values'])
plt.ylabel('Contract Value in Billions')
```

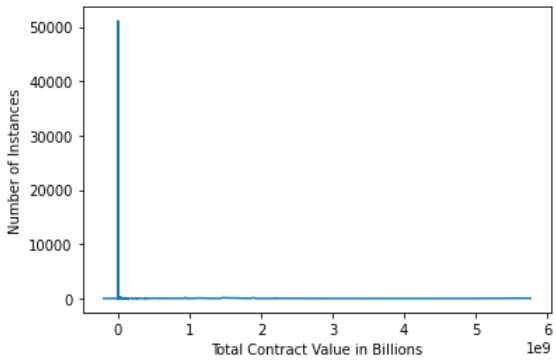
Out[ ]:
```
Text(0, 0.5, 'Contract Value in Billions')
```



In [ ]:
```
totalValue = contractHistory['total-contract-value'].value_counts().rename_axis('unique_values').reset_index(name='counts')
totalValue = pd.DataFrame(totalValue)
totalValue = totalValue.sort_values(by='unique_values')
```

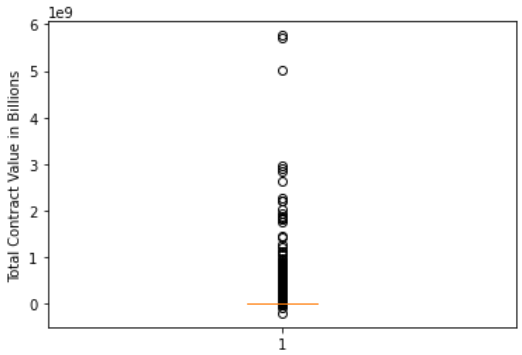
In [ ]:
```
plt.plot(totalValue['unique_values'],totalValue['counts'])
plt.ylabel('Number of Instances')
plt.xlabel('Total Contract Value in Billions')
```

Out[ ]:
```
Text(0.5, 0, 'Total Contract Value in Billions')
```



In [ ]:
```
plt.boxplot(totalValue['unique_values'])
plt.ylabel('Total Contract Value in Billions')
```

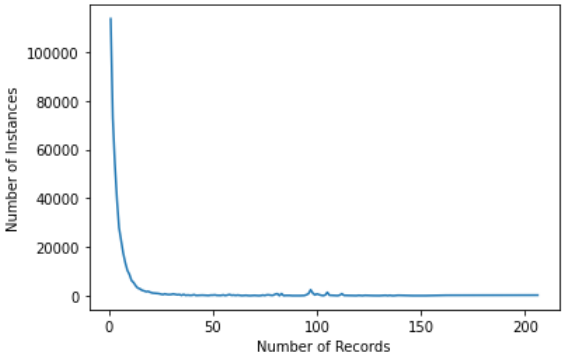
Out[ ]:
```
Text(0, 0.5, 'Total Contract Value in Billions')
```



In [ ]:
```
noRecord = contractHistory['number-records'].value_counts().rename_axis('unique_values').reset_index(name='counts')
noRecord = pd.DataFrame(noRecord)
noRecord = noRecord.sort_values(by='unique_values')
```

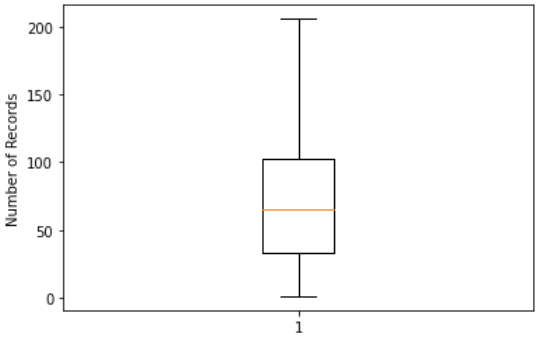
In [ ]:
```
plt.plot(noRecord['unique_values'],noRecord['counts'])
plt.ylabel('Number of Instances')
plt.xlabel('Number of Records')
```

Out[ ]:
```
Text(0.5, 0, 'Number of Records')
```



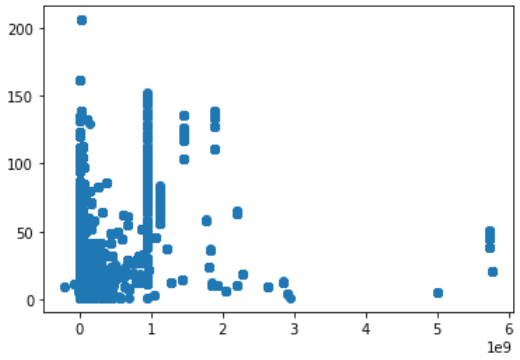
In [ ]:
```
plt.boxplot(noRecord['unique_values'])
plt.ylabel('Number of Records')
```

Out[ ]:
```
Text(0, 0.5, 'Number of Records')
```



In [ ]:
```
plt.scatter(contractHistory['total-contract-value'],contractHistory['number-records'])
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x181d4ca2ac0>
```



Correlation Analysis

In [ ]:
```
corrMatrix = contractHistory.corr()
sn.heatmap(corrMatrix, annot=True)
plt.show()
```

Removing Attributes

Duplicate attributes were removed, barring GSIN and trade agreements, since this information might help with later analysis. Procurement entity information was removed since it showed one value.

In [ ]:
```
editCH = contractHistory.drop(['gsin-description_fr','competitive-tender_fr','limited-tender-reason','limited-tender-reason-description_fr','solici
       'procurement-entity-name_fr'], axis = 1)
```

Testing for Impact of Duplicate Contract Number Instances

Reviewing to determine whether duplicate contact numbers result in duplicate information elsewhere.

In [ ]:
```
editCH.insert(27, 'contract-number supplier test',editCH['contract-number'] + editCH['supplier-standardized-name'])
```

In [ ]:
```
editCH['contract-number supplier test'].value_counts()
```

Out[ ]:
```
E0208-150548/001/PWZWRIGHT CONSTRUCTION                                                                                  206
EN578-110558/001/XLWOLTERS KLUWER LTD / WOLTERS KLUWER LIMITEE                                                           162
EP008-112560/001/GCBROOKFIELD GLOBAL INTEGRATED SOLUTIONS CANADA LP/BROOKFIELD SOLUTIONS GLOBALES INTEGREES CANADA SEC    152
EW038-140681/001/PWUTRI CITY CANADA INC                                                                                  139
EP008-112560/004/GCBROOKFIELD GLOBAL INTEGRATED SOLUTIONS CANADA LP/BROOKFIELD SOLUTIONS GLOBALES INTEGREES CANADA SEC    139
                                                                                                                        ...
W8160-140026/001/PICEDROM-SNI INC                                                                                          1
08324-140303/001/EJONX ENTERPRISE SOLUTIONS LTD                                                                            1
W8486-096164/005/HSB S F (BUSINESS SOLUTIONS FASTENERS) INTERNATIONAL INC                                                  1
E60LP-100002/088/LPSHERATON VANCOUVER GUILDFORD                                                                            1
U4030-221421/001/HNSPEAG SCHMID & PARTNER ENGINEERING AG                                                                   1
Name: contract-number supplier test, Length: 200644, dtype: int64
```

In [ ]:
```
editCH['contract-number'].describe()
```

Out[ ]:
```
count                  441210
unique                 199675
top       E0208-150548/001/PWZ
freq                      206
Name: contract-number, dtype: object
```

In [ ]:
```
editCH['contract-number supplier test'].describe()
```

Out[ ]:
```
count                                    441210
unique                                   200644
top       E0208-150548/001/PWZWRIGHT CONSTRUCTION
freq                                        206
Name: contract-number supplier test, dtype: object
```

Removing Duplicate Contract Number Instances

Since the test above indicates that there are only 969 unique instances where the supplier changes within the same contract number, representing only 0.485% of the contract numbers. The duplicate instances of contract numbers to reflect amendments were removed, as well as the contract value and amendment number, which are better reflected by the total contrat value and number of records attributes. The remaining instances reflect the first award dates of the contract numbers to capture the initial win of the contract.

In [ ]:
```
editCH.sort_values(['contract-number','award-date'], ascending=True, inplace=True)
editCH.insert(28, 'duplicate contract check', editCH.duplicated(subset='contract-number'))
editCH = editCH[editCH['duplicate contract check'] != True]
editCH.shape
```

Out[ ]:
```
(199675, 29)
```

In [ ]:
```
editCH = editCH.drop(['amendment-number', 'contract-value', 'duplicate contract check', 'contract-number supplier test'], axis = 1)
editCH.shape
```

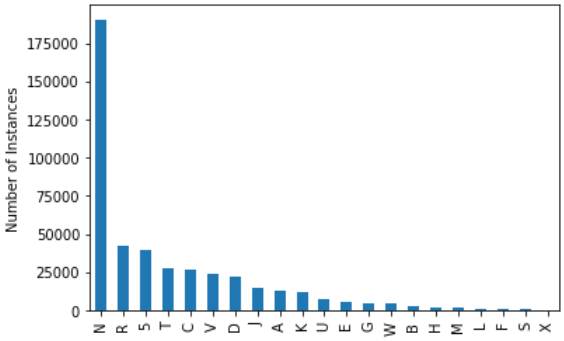
Out[ ]:
```
(199675, 25)
```

Creating a Parent GSIN Category

Given the high volume of unique GSIN codes, and the ability to categorize the code within a parent company by the first letter, a GSIN category attribute has been created.

In [ ]:
```
editCH.insert(5, 'gsin-category', editCH['gsin'].str[:1])
editCH['gsin-category'].value_counts().plot(kind = 'bar')
plt.ylabel('Number of Instances')
```

Out[ ]:
```
Text(0, 0.5, 'Number of Instances')
```



Removing Limited Tender Reason

The information in the editCH dataset is generally filled in. However, a category with a significant amount (72.5%) missing values is the the limited tender reason column, which is a near exact match to the number of non-competitive tenders. Given the sparce data in this column, it has been removed from the dataset.

In [ ]:
```
editCH['contract-number'].describe()
```

Out[ ]:
```
count                  199675
unique                 199675
top       01005-010324/009/WPG
freq                        1
Name: contract-number, dtype: object
```

In [ ]:
```
editCH['limited-tender-reason-description_en'].describe()
```

Out[ ]:
```
count               54900
unique                 16
top       Exclusive Rights
freq                30599
Name: limited-tender-reason-description_en, dtype: object
```

In [ ]:
```
(199675-54900)/199675*100
```

Out[ ]:
```
72.50532114686365
```

In [ ]:
```
editCH['competitive-tender_en'].value_counts()
```

Out[ ]:
```
Yes    144768
No      54907
Name: competitive-tender_en, dtype: int64
```

In [ ]:
```
editCH = editCH.drop(['limited-tender-reason-description_en'], axis = 1)
```

Managing Missing Categorical Values

The the following items have lower volumes of missing data: trade agreement (0.47%), supplier details such as city (0.23%), province/state (3.15%), postal code (0.96%), organization employee count (0.0005%), and contracting entity office name (2.66%) and details such as address street 1 (0.0015%), address street 2 (26.98%), province/state (0.075%), postal code (0.002%), and country (0.0015%). Since the freqnency of missing information is limited in all attributes except for address street 2, the rows containing missing data were removed. The attribute for address street 2 was due to high missing volume. However, province/state missing items were not removed since the missing cells reflect suppliers from other countries.

In [ ]:
```
#this cell was repeated for all attributes with missing values to assess impact
editCH['gsin-description_en'].describe()
```

Out[ ]:
```
count                                        199675
unique                                         5164
top       Hotels, Motels and Commercial Accommodation
freq                                          16657
Name: gsin-description_en, dtype: object
```

In [ ]:
```
editCH['contract-number'].describe()
```

Out[ ]:
```
count                  199675
unique                 199675
top       01005-010324/009/WPG
freq                        1
Name: contract-number, dtype: object
```

In [ ]:
```
#this cell was repeated for all attributes with missing values to assess impact
(199675-199672)/199675*100
```

Out[ ]:
```
0.0015024414673844998
```

In [ ]:

```
editCH = editCH.drop(['contracting-address-street-2'], axis = 1)
```

In [ ]:
```
#editCH.dropna(subset = ['gsin-description_en', 'end-user-entity-top', 'trade-agreement', 'supplier-address-city', 'supplier-address-postal-code',
```

In [ ]:
```
editCH.columns
```

Out[ ]:
```
Index(['contract-number', 'award-date', 'expiry-date', 'gsin',
       'gsin-description_en', 'gsin-category', 'competitive-tender_en',
       'solicitation-procedure-description_en', 'trade-agreement',
       'supplier-standardized-name', 'supplier-address-city',
       'supplier-address-prov-state', 'supplier-address-postal-code',
       'supplier-address-country', 'organization-employee-count_en',
       'total-contract-value', 'number-records', 'end-user-entity_en',
       'contracting-entity-office-name_en', 'contracting-address-street-1',
       'contracting-address-city', 'contracting-address-prov-state',
       'contracting-address-postal-code', 'contracting-address-country'],
      dtype='object')
```

In [ ]:
```
editCH.dropna(subset = ['contract-number', 'award-date', 'expiry-date', 'gsin', 'gsin-description_en', 'gsin-category', 'competitive-tender_en', 's
```

In [ ]:
```
editCH.shape
```

Out[ ]:
```
(194135, 24)
```

Creating Time Series for Award Date Attribute

The expiry date was removed after review of the data.

In [ ]:
```
editCH['award year']=[d.split('-')[0] for d in editCH['award-date']]
editCH['award month']=[d.split('-')[1] for d in editCH['award-date']]
editCH['award day']=[d.split('-')[2] for d in editCH['award-date']]
```

In [ ]:
```
editCH['award year'] = pd.to_numeric(editCH['award year'])
```

In [ ]:
```
awardYear = (editCH['award year'] - editCH['award year'].min()) / (editCH['award year'].max()-editCH['award year'].min())
editCH.insert(25, 'award-year-normalized',awardYear)
```

In [ ]:
```
editCH.insert(24, 'days-since-first-award', 'None')
```

In [ ]:
```
from datetime import datetime

editCH['award-date'] = editCH['award-date'].apply(pd.to_datetime)
editCH['days-since-first-award'] = (editCH['award-date'] - editCH['award-date'].min()).dt.days
editCH['days-since-first-award']
```

Out[ ]:
```
22402       89
22486      127
22488      259
279171      63
204467     243
          ...
47968     1121
91429     1145
74499     1134
285173    1175
167242     152
Name: days-since-first-award, Length: 194135, dtype: int64
```

In [ ]:
```
normalizedAwardDays = (editCH['days-since-first-award'] - editCH['days-since-first-award'].min()) / (editCH['days-since-first-award'].max()-editCH[
editCH.insert(25, 'days-since-first-award-normalized',normalizedAwardDays)
```

Numeric Attributes

Total Contract Value

This data has a right skew, with negative values, a significant spike in values at 0 and 25000, and a long right tail with outliers.

In [ ]:
```
editCH['total-contract-value'].value_counts()
```

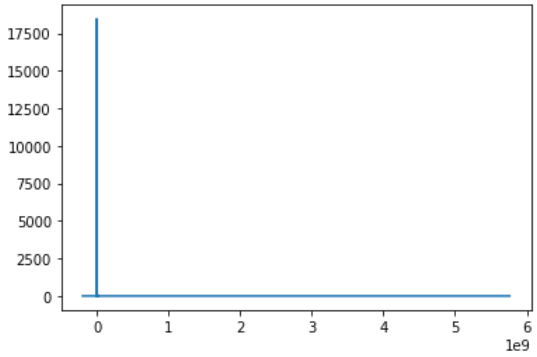
Out[ ]:
```
0.0          18461
25000.0      10848
1.0            982
100000.0       758
26250.0        697
             ...
```

```
3170196.0        1
615523.0         1
4769883.0        1
176914.0         1
3999.0           1
Name: total-contract-value, Length: 96040, dtype: int64
```

In [ ]:
```
totalContractValue = editCH['total-contract-value'].value_counts().rename_axis('unique_values').reset_index(name='counts')
totalContractValue = pd.DataFrame(totalContractValue)
totalContractValue = totalContractValue.sort_values(by = 'unique_values')
```

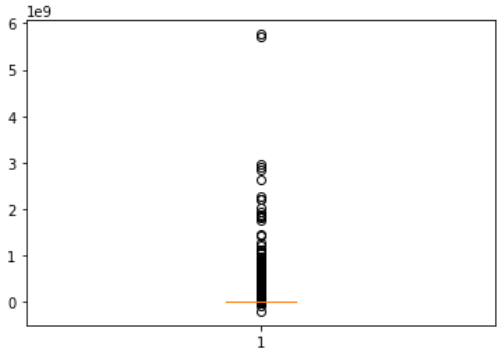
In [ ]:
```
plt.plot(totalContractValue['unique_values'],totalContractValue['counts'])
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x181f2220a60>]
```



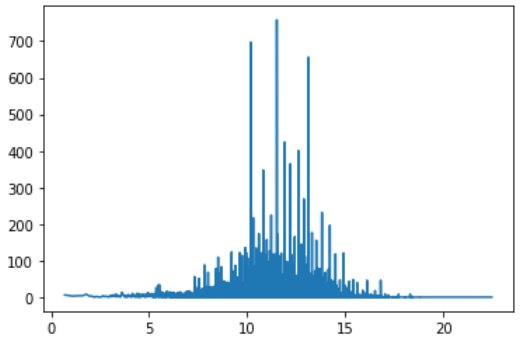
In [ ]:
```
plt.boxplot(totalContractValue['unique_values'])
```

Out[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x181f22421c0>,
  <matplotlib.lines.Line2D at 0x181f2242550>],
 'caps': [<matplotlib.lines.Line2D at 0x181f2242910>,
  <matplotlib.lines.Line2D at 0x181f2242ca0>],
 'boxes': [<matplotlib.lines.Line2D at 0x181f223fdf0>],
 'medians': [<matplotlib.lines.Line2D at 0x181f2243070>],
 'fliers': [<matplotlib.lines.Line2D at 0x181f2243400>],
 'means': []}
```



In [ ]:
```
test = totalContractValue[['unique_values', 'counts']]
test = test[test['unique_values'] > 1]
test = test[test['unique_values'] != 25000]
plt.plot(np.log(test['unique_values']),test['counts'])
```

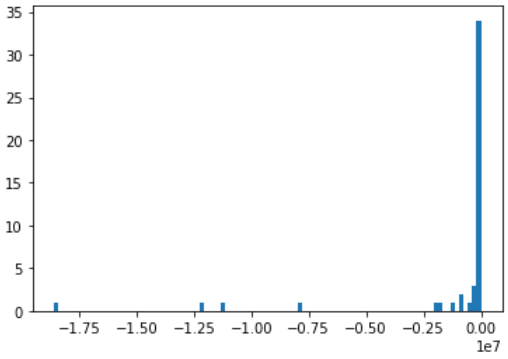
Out[ ]:
```
[<matplotlib.lines.Line2D at 0x182079e2fd0>]
```



In [ ]:
```
test2 = totalContractValue[['unique_values', 'counts']]
test2 = test2[test2['unique_values'] < 0]
test2 = test2[test2['unique_values'] > -25000000]
plt.hist(test2['unique_values'], bins = 100)
```

```
(array([ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
```

Out[ ]:
```
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,
        1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
        0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  1.,
        0.,  0.,  1.,  0.,  2.,  0.,  1.,  3., 34.]),
array([-1.85800910e+07, -1.83942901e+07, -1.82084892e+07, -1.80226883e+07,
       -1.78368874e+07, -1.76510865e+07, -1.74652856e+07, -1.72794847e+07,
       -1.70936838e+07, -1.69078829e+07, -1.67220820e+07, -1.65362811e+07,
       -1.63504802e+07, -1.61646793e+07, -1.59788784e+07, -1.57930775e+07,
       -1.56072766e+07, -1.54214757e+07, -1.52356748e+07, -1.50498739e+07,
       -1.48640730e+07, -1.46782721e+07, -1.44924712e+07, -1.43066703e+07,
       -1.41208694e+07, -1.39350685e+07, -1.37492676e+07, -1.35634667e+07,
       -1.33776658e+07, -1.31918649e+07, -1.30060640e+07, -1.28202631e+07,
       -1.26344622e+07, -1.24486613e+07, -1.22628604e+07, -1.20770595e+07,
       -1.18912586e+07, -1.17054577e+07, -1.15196568e+07, -1.13338559e+07,
       -1.11480550e+07, -1.09622541e+07, -1.07764532e+07, -1.05906523e+07,
       -1.04048514e+07, -1.02190505e+07, -1.00332496e+07, -9.84744870e+06,
       -9.66164780e+06, -9.47584690e+06, -9.29004600e+06, -9.10424510e+06,
       -8.91844420e+06, -8.73264330e+06, -8.54684240e+06, -8.36104150e+06,
       -8.17524060e+06, -7.98943970e+06, -7.80363880e+06, -7.61783790e+06,
       -7.43203700e+06, -7.24623610e+06, -7.06043520e+06, -6.87463430e+06,
       -6.68883340e+06, -6.50303250e+06, -6.31723160e+06, -6.13143070e+06,
       -5.94562980e+06, -5.75982890e+06, -5.57402800e+06, -5.38822710e+06,
       -5.20242620e+06, -5.01662530e+06, -4.83082440e+06, -4.64502350e+06,
       -4.45922260e+06, -4.27342170e+06, -4.08762080e+06, -3.90181990e+06,
       -3.71601900e+06, -3.53021810e+06, -3.34441720e+06, -3.15861630e+06,
       -2.97281540e+06, -2.78701450e+06, -2.60121360e+06, -2.41541270e+06,
       -2.22961180e+06, -2.04381090e+06, -1.85801000e+06, -1.67220910e+06,
       -1.48640820e+06, -1.30060730e+06, -1.11480640e+06, -9.29005500e+05,
       -7.43204600e+05, -5.57403700e+05, -3.71602800e+05, -1.85801900e+05,
       -1.00000000e+00]),
<BarContainer object of 100 artists>)
```

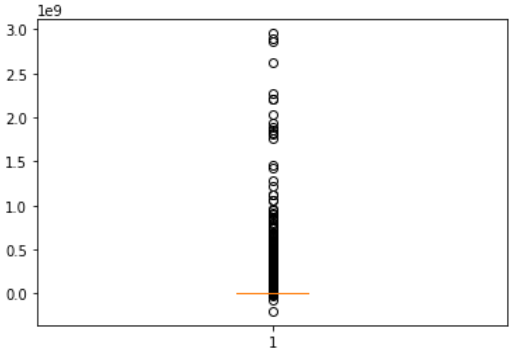


In [ ]:
```
#This data has a lognormal distribution for the positive values when 0, 1, and 25000 are removed
```

In [ ]:
```
#removing two significant outliers
remove = editCH[editCH['total-contract-value'] >= 4000000000].index
editCH.drop(remove, inplace = True)
```

In [ ]:
```
plt.boxplot(editCH['total-contract-value'])
```

Out[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x181f1b9bf70>,
  <matplotlib.lines.Line2D at 0x181f1ba0340>],
 'caps': [<matplotlib.lines.Line2D at 0x181f1ba06d0>,
  <matplotlib.lines.Line2D at 0x181f1ba0a60>],
 'boxes': [<matplotlib.lines.Line2D at 0x181f1b9bbe0>],
 'medians': [<matplotlib.lines.Line2D at 0x181f1ba0df0>],
 'fliers': [<matplotlib.lines.Line2D at 0x181f1ba31c0>],
 'means': []}
```



In [ ]:
```
normalizedContractValue = (editCH['total-contract-value'] - editCH['total-contract-value'].min()) / (editCH['total-contract-value'].max()-editCH['t
editCH.insert(16, 'total-contract-value-normalized',normalizedContractValue)
```

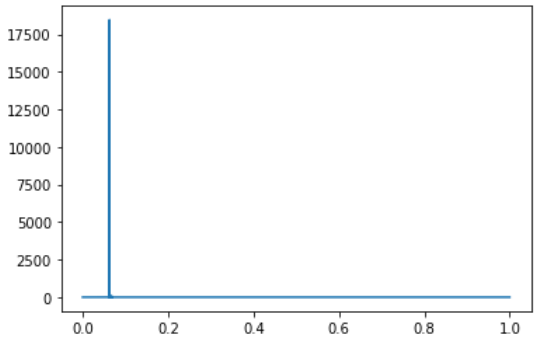
In [ ]:
```
contractValueNorm = editCH['total-contract-value-normalized'].value_counts().rename_axis('unique_values').reset_index(name='counts')
contractValueNorm = pd.DataFrame(contractValueNorm)
contractValueNorm = contractValueNorm.sort_values(by='unique_values')
print(contractValueNorm)
```

```
       unique_values  counts
46268       0.000000       1
46746       0.038139       1
46156       0.057092       1
54049       0.059099       1
46246       0.059409       1
...              ...     ...
56932       0.785771       1
44460       0.896624       1
92348       0.969658       1
44286       0.982991       1
93428       1.000000       1

[96038 rows x 2 columns]
```

In [ ]:
```
plt.plot(contractValueNorm['unique_values'],contractValueNorm['counts'])
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x181eff11b50>]
```



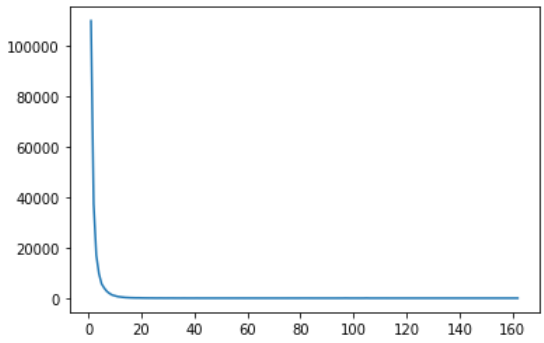
Number of Records

In [ ]:
```
editCH['number-records'].value_counts()
```

Out[ ]:
```
1      110041
2       37463
3       17048
4        9743
5        5698
        ...
103         1
67          1
135         1
123         1
64          1
Name: number-records, Length: 105, dtype: int64
```

In [ ]:
```
noRecord = editCH['number-records'].value_counts().rename_axis('records').reset_index(name='counts')
noRecord = pd.DataFrame(noRecord)
noRecord = noRecord.sort_values(by='records')
plt.plot(noRecord['records'],noRecord['counts'])
```

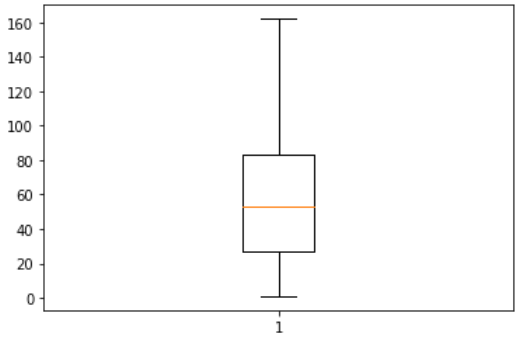
Out[ ]:
```
[<matplotlib.lines.Line2D at 0x181eff3e040>]
```



In [ ]:
```
plt.boxplot(noRecord['records'])
```

Out[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x181eff695b0>,
  <matplotlib.lines.Line2D at 0x181eff69940>],
 'caps': [<matplotlib.lines.Line2D at 0x181eff69cd0>,
  <matplotlib.lines.Line2D at 0x181eff6d0a0>],
 'boxes': [<matplotlib.lines.Line2D at 0x181eff69220>],
 'medians': [<matplotlib.lines.Line2D at 0x181eff6d430>],
```

```
'fliers': [<matplotlib.lines.Line2D at 0x181eff6d7c0>],
'means': []}
```



In [ ]:
```
#This data has an exponential distibution
#Since number of records does not have material impact on the outcome of the supplier award, it is being removed from this analysis
#noRecordNorm = (editCH['number-records'] - editCH['number-records'].min()) / (editCH['number-records'].max()-editCH['number-records'].min())
#editCH.insert(17, 'number-records-normalized',noRecordNorm)
```

In [ ]:
```
editCH = editCH.drop(['number-records'], axis = 1)
```

In [ ]:
```
#editCH = editCH.drop(['award year', 'award month', 'award day', 'expiry year', 'expiry month', 'expiry day'], axis = 1)
```

In [ ]:
```
editCH.describe()
```

Out[ ]:

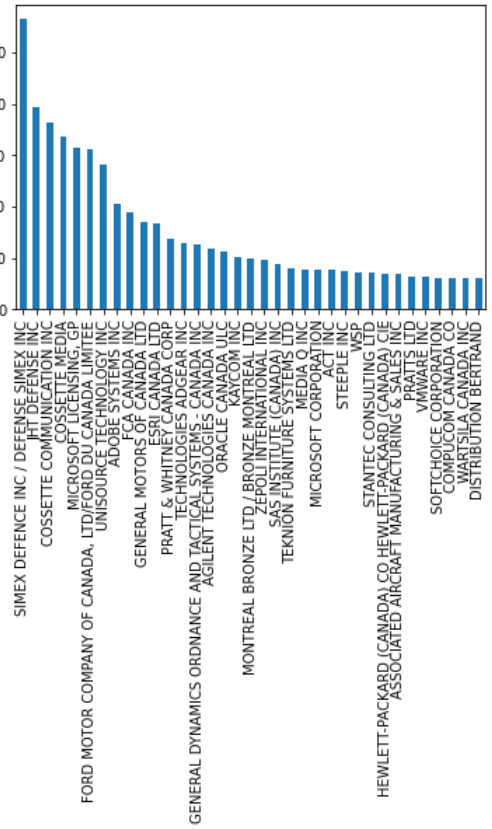
	total-contract-value	total-contract-value-normalized	days-since-first-award	days-since-first-award-normalized	award year	award-year-normalized
count	1.941330e+05	194133.000000	194133.000000	194133.000000	194133.000000	194133.000000
mean	1.149437e+06	0.063351	1607.475597	0.346289	2012.982507	0.331876
std	2.434127e+07	0.007722	1310.866238	0.282393	3.570864	0.297572
min	-1.985487e+08	0.000000	0.000000	0.000000	2009.000000	0.000000
25%	1.128000e+04	0.062989	479.000000	0.103188	2010.000000	0.083333
50%	4.320000e+04	0.063000	1202.000000	0.258940	2012.000000	0.250000
75%	2.014690e+05	0.063050	2582.000000	0.556226	2016.000000	0.583333
max	2.953723e+09	1.000000	4642.000000	1.000000	2021.000000	1.000000

Creating the Dependent Variable

To create a binary dependent variable that will determine whether a supplier will be the supplier on a contract, a new column need sto be created that flags all instances where the supplier is named and turns those instances into a Yes. All other instances will be No. The supplier to be used in this analysis is STANTEC. Note that this attribute is messy, meaning that STANTEC is named in multiple unique instances.

In [ ]:
```
topSuppliers = editCH['supplier-standardized-name'].value_counts()
topSuppliers = topSuppliers[topSuppliers > 300]
topSuppliers.plot(kind = 'bar')
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
editCH.insert(9,'stantec-supplier','no')
```

In [ ]:
```
editCH['stantec-supplier'] = np.where(editCH['supplier-standardized-name'].str.contains('STANTEC'), 'yes', editCH['stantec-supplier'])
```

In [ ]:
```
editCH['stantec-supplier'].value_counts()
```

Out[ ]:
```
no     193551
yes       582
Name: stantec-supplier, dtype: int64
```

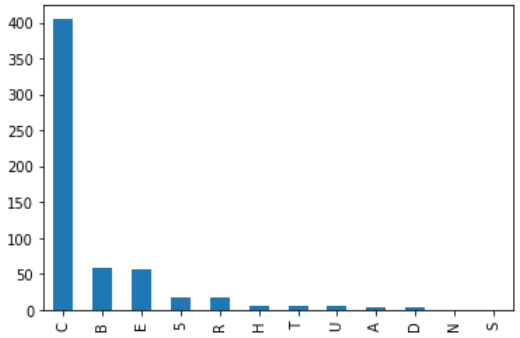
Look at Subsets & How Attributes Link to the Class

Value counts were run for all attributes anything that looked distinct is included in the code below.

In [ ]:
```
stantecData = editCH[editCH['stantec-supplier'] == 'yes']
```

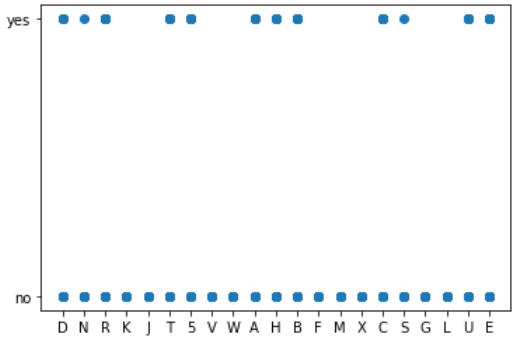
In [ ]:
```
stantecData['gsin-category'].value_counts().plot(kind = 'bar')
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
plt.scatter(editCH['gsin-category'],editCH['stantec-supplier'])
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x181f3f2caf0>
```



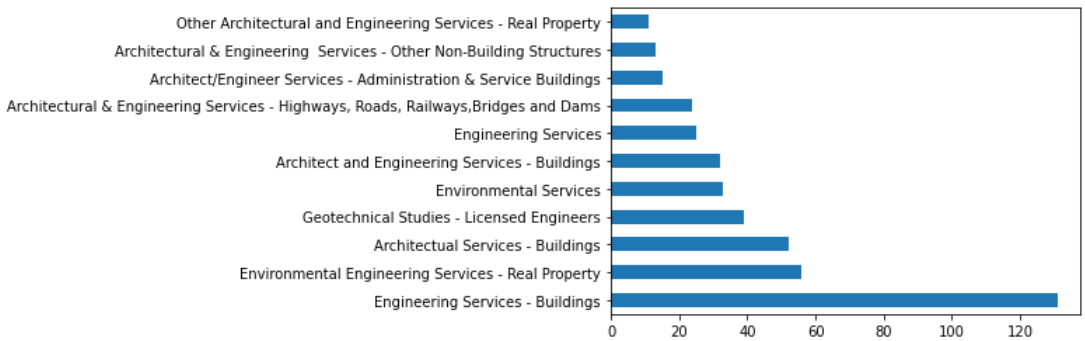
In [ ]:
```
yesGsin = pd.DataFrame(stantecData['gsin-category'].value_counts().rename_axis('gsin').reset_index(name='stantec count'))
allGsin = pd.DataFrame(editCH['gsin-category'].value_counts().rename_axis('gsin').reset_index(name='all count'))
gsinMerge = pd.merge(allGsin, yesGsin, how = 'outer')
gsinMerge
#gsinMerge.plot(kind = 'bar')
```

Out[ ]:

	gsin	all count	stantec count
0	N	99022	1.0
1	V	18867	NaN
2	R	13485	18.0
3	T	12615	5.0
4	5	10881	18.0
5	C	7882	405.0
6	D	7768	4.0
7	J	5996	NaN
8	A	4296	4.0
9	K	3479	NaN
10	U	2656	5.0
11	E	1808	56.0
12	W	1672	NaN
13	B	1211	59.0
14	G	959	NaN
15	H	598	6.0
16	L	273	NaN
17	F	213	NaN
18	S	194	1.0
19	M	140	NaN
20	X	118	NaN

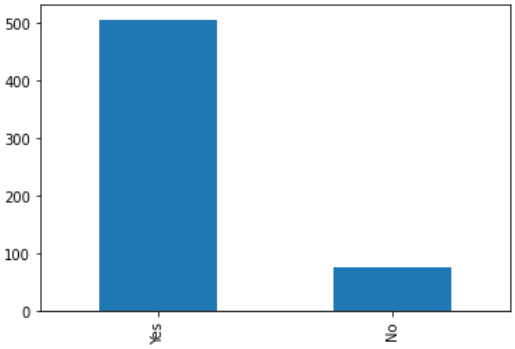
In [ ]:
```
gsinStantec = stantecData['gsin-description_en'].value_counts()
gsinStantec = gsinStantec[gsinStantec > 10]
gsinStantec.plot(kind = 'barh')
```

Out[ ]:
```
<AxesSubplot:>
```



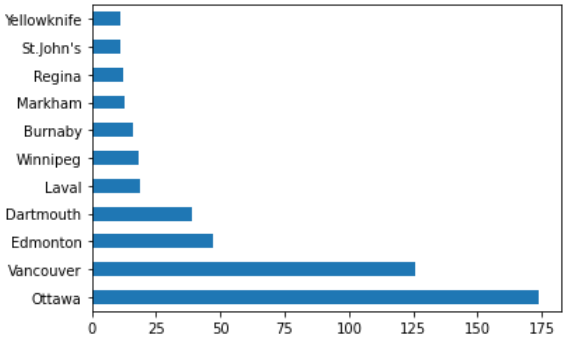
In [ ]:
```
stantecData['competitive-tender_en'].value_counts().plot(kind = 'bar')
```

Out[ ]:
```
<AxesSubplot:>
```



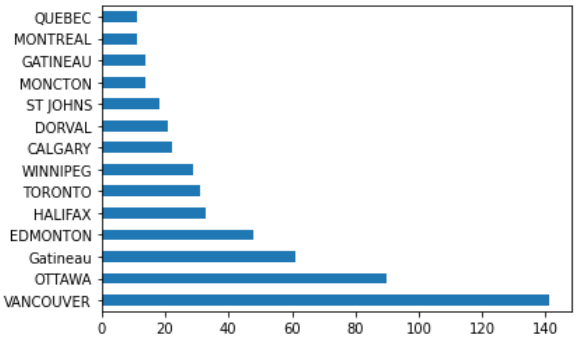
In [ ]:
```
officeStantec = stantecData['supplier-address-city'].value_counts()
officeStantec = officeStantec[officeStantec > 10]
officeStantec.plot(kind = 'barh')
```

Out[ ]:
```
<AxesSubplot:>
```



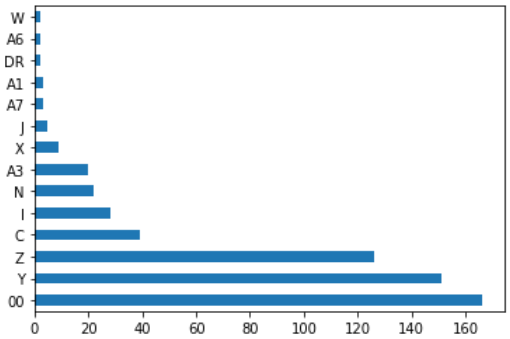
In [ ]:
```
locateStantec = stantecData['contracting-address-city'].value_counts()
locateStantec = locateStantec[locateStantec > 10]
locateStantec.plot(kind = 'barh')
```

Out[ ]:
```
<AxesSubplot:>
```



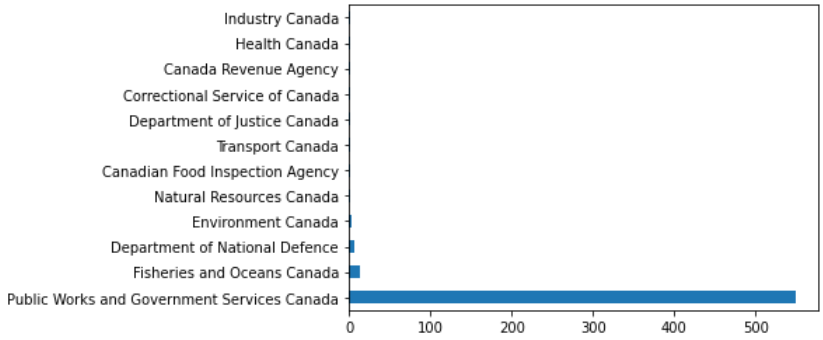
In [ ]:
```
tradeStantec = stantecData['trade-agreement'].value_counts()
tradeStantec = tradeStantec[tradeStantec > 1]
tradeStantec.plot(kind = 'barh')
```

Out[ ]:
```
<AxesSubplot:>
```



In [ ]:
```
stantecData['end-user-entity_en'].value_counts().plot(kind = 'barh')
```

Out[ ]:
```
<AxesSubplot:>
```



Removing Non-Applicable GSIN Categories

Based on the analysis above, the GSIN Categories V, J, K, W, G, L, F, M, X have no Stantec supplier counts, and GSIN Category N only has one our of 97533.

In [ ]:
```
editCH.insert(27, 'remove-gsin', 'no')
```

In [ ]:
```
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('V'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('J'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('K'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('W'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('G'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('L'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('F'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('M'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('X'), 'yes', editCH['remove-gsin'])
editCH['remove-gsin'] = np.where(editCH['gsin-category'].str.contains('N'), 'yes', editCH['remove-gsin'])
```

In [ ]:
```
editCH['remove-gsin'].value_counts()
```

Out[ ]:
```
yes    130739
no      63394
Name: remove-gsin, dtype: int64
```

In [ ]:
```
index_names = editCH[editCH['remove-gsin'] == 'yes'].index
editCH.drop(index_names, inplace = True)
```

Reducing the Number of Unique End User Departments, GSIN Codes, Contracting Cities, Contracting Postal Code

In [ ]:
```
editCH.insert(5,'gsin-description-top', editCH['gsin-description_en'])
editCH.insert(19,'end-user-entity-top', editCH['end-user-entity_en'])
editCH.insert(23,'contracting-address-city-top', editCH['contracting-address-city'])
editCH.insert(26,'contracting-address-postal-top', editCH['contracting-address-postal-code'])
```

In [ ]:
```
frequencies = editCH['gsin-description-top'].value_counts(normalize=True, ascending=True)
mapping = editCH['gsin-description-top'].map(frequencies)
editCH['gsin-description-top'] = editCH['gsin-description-top'].mask(mapping < 0.005, 'Other')
editCH['gsin-description-top'].value_counts()
```

Out[ ]:
```
Other                                                                                                 23486
Informatics Professional Services                                                                      5590
Human Resource Services, Business Consulting/Change Management; Project Management Services           3774
Advertising Media Planning/Buying                                                                      3376
Temporary Help Services, General Office Support                                                        1658
Translation Services                                                                                   1615
Engineering Services - Buildings                                                                       1427
Architectual Services - Buildings                                                                      1344
Information Products                                                                                   1300
Construction of Other Buildings                                                                        1278
Audio Visual Production Services                                                                       1054
Miscellaneous Business Services                                                                         994
Military (R&D)                                                                                          953
Waterways, Harbours, Dams and Other Water Works                                                         944
Business Services                                                                                       872
Product/Material - Design, Development, Formulation, Modification: Science and Technology Related (R&D)  797
Environmental Services                                                                                  762
Architectural & Engineering Services - Highways, Roads, Railways,Bridges and Dams                       719
Advertising Related Services                                                                            641
Media Monitoring Services                                                                               605
Engineering Services                                                                                    602
Architect/Engineer Services - Administration & Service Buildings                                        596
Video Production Services                                                                               586
Public Opinion Research - Quantitative                                                                  581
Communication Promotional Material (including printing and identification)                              501
End-to-End Learning Services (Excludes COTS Training)                                                   497
General Marine Construction Work                                                                        496
```

```
Astronautics (R&D)                                                                473
Electrical Installations and Major Repairs                                        461
Press Release Distribution                                                        447
Heating, Ventilation and Air Conditioning Maintenance & Inspection Services       403
Public Opinion Research - Quantitative & Qualitative                              392
Technical / Vocational Training                                                   375
Internal and External Audits (Supply Arrangement PASS)                            374
Publications                                                                      362
Architect and Engineering Services - Buildings                                    362
Information Retrieval Services, Database                                          360
Marine Architect and Engineering Services                                         349
General Office Help                                                               338
Highways, Roads, Railways,  Airfield Runways                                      338
Public Opinion Research - Qualitative                                             335
Language Interpretation Services                                                  329
Advertising Creative Services                                                     325
General Contractor Services, Not Elsewhere Specified                              323
Name: gsin-description-top, dtype: int64
```

In [ ]:

```
frequencies = editCH['end-user-entity-top'].value_counts(normalize=True, ascending=True)
mapping = editCH['end-user-entity-top'].map(frequencies)
editCH['end-user-entity-top'] = editCH['end-user-entity-top'].mask(mapping < 0.005, 'Other')
editCH['end-user-entity-top'].value_counts()
```

Out[ ]:

```
Public Works and Government Services Canada              35854
Department of National Defence                           10171
Other                                                     4521
Parks Canada                                              1647
Fisheries and Oceans Canada                               1325
Health Canada                                             1154
Natural Resources Canada                                   935
Employment and Social Development Canada                   883
Canadian Space Agency                                      857
Agriculture and Agri-Food Canada                           716
Industry Canada                                            705
Citizenship and Immigration Canada                         669
Environment Canada                                         606
Transport Canada                                           570
Foreign Affairs, Trade And Development (Department Of)     545
Veterans Affairs Canada                                    481
Canada Revenue Agency                                      456
Correctional Service of Canada                             455
Royal Canadian Mounted Police                              446
Aboriginal Affairs & Northern Development Canada           398
Name: end-user-entity-top, dtype: int64
```

In [ ]:

```
frequencies = editCH['contracting-address-city-top'].value_counts(normalize=True, ascending=True)
mapping = editCH['contracting-address-city-top'].map(frequencies)
editCH['contracting-address-city-top'] = editCH['contracting-address-city-top'].mask(mapping < 0.01, 'Other')
editCH['contracting-address-city-top'].value_counts()
```

Out[ ]:

```
OTTAWA           17058
GATINEAU         10250
Other             8536
Gatineau          7585
VANCOUVER         3243
MONTREAL          1889
HALIFAX           1880
EDMONTON          1753
QUEBEC            1665
WINNIPEG          1447
MONCTON           1277
TORONTO           1197
ST JOHNS          1004
CHARLOTTETOWN      917
MISSISSAUGA        811
CALGARY            784
ST HUBERT          741
BORDEN             683
DARTMOUTH          674
Name: contracting-address-city-top, dtype: int64
```

In [ ]:

```
frequencies = editCH['contracting-address-postal-top'].value_counts(normalize=True, ascending=True)
mapping = editCH['contracting-address-postal-top'].map(frequencies)
editCH['contracting-address-postal-top'] = editCH['contracting-address-postal-top'].mask(mapping < 0.01, 'Other')
editCH['contracting-address-postal-top'].value_counts()
```

Out[ ]:

```
Other     24683
K1A0S5    17645
K1A0K2     2932
V6Z0B9     2605
M2N6A6     1596
T5J1S6     1535
H5A1L6     1365
K1A0K9     1294
B3J3C9     1141
E1C1H1     1105
R3B0T6     1026
K1A0H4      990
K1A0Z4      930
```

```
J3Y8Y9      854
L5B2N5      808
K1A0J9      789
A1C5T2      771
L0M1C0      683
K1A1L1      642
Name: contracting-address-postal-top, dtype: int64
```

Attribute Selection

In [ ]:
```
modelCH = editCH.drop(['contract-number', 'award-date', 'expiry-date', 'gsin', 'gsin-description_en', 'supplier-standardized-name', 'supplier-addre
```

In [ ]:
```
modelCH.columns
```

Out[ ]:
```
Index(['gsin-description-top', 'gsin-category', 'competitive-tender_en',
       'solicitation-procedure-description_en', 'trade-agreement',
       'stantec-supplier', 'total-contract-value-normalized',
       'end-user-entity-top', 'contracting-address-city-top',
       'contracting-address-prov-state', 'contracting-address-postal-top',
       'days-since-first-award-normalized', 'award-year-normalized',
       'award month'],
      dtype='object')
```

Clustering

In [ ]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5)
trainReshape = modelCH['total-contract-value-normalized'].values.reshape(-1, 1)
clusterCV = kmeans.fit(trainReshape)
clusterCV.cluster_centers_
```

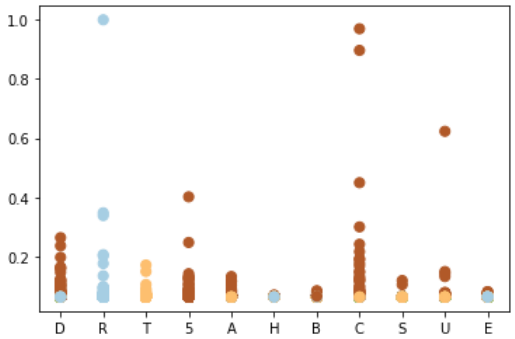
Out[ ]:
```
array([[0.06321759],
       [0.95542732],
       [0.24438324],
       [0.12491418],
       [0.49130407]])
```

In [ ]:
```
from kmodes.kprototypes import KPrototypes
from kmodes.kmodes import KModes
```

In [ ]:
```
kproto = KPrototypes(n_clusters = 3)
trainSet = modelCH[['gsin-category','total-contract-value-normalized']]
cluster = kproto.fit(trainSet, categorical = [0])
yPred = kproto.predict(trainSet, categorical = [0])
cluster.cluster_centroids_
```

Out[ ]:
```
array([['0.06327257371842605', 'R'],
       ['0.0630452900093655', 'T'],
       ['0.064347374691316', '5']], dtype='<U32')
```

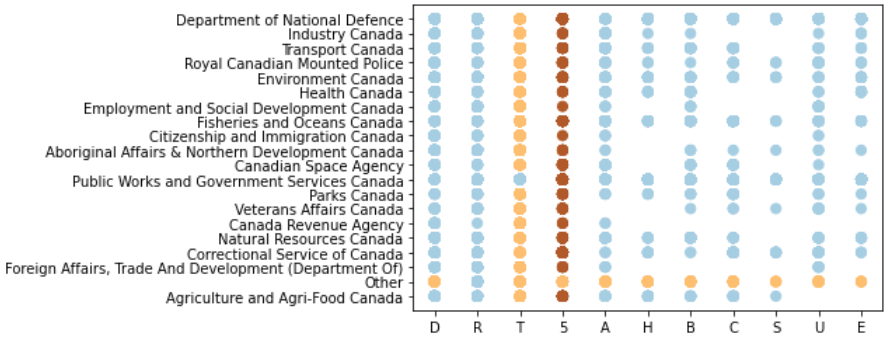
In [ ]:
```
plt.scatter(trainSet['gsin-category'], trainSet['total-contract-value-normalized'], s = 50, c = yPred, cmap = plt.cm.Paired)
plt.show()
```



In [ ]:
```
kmode = KModes(n_clusters = 3)
trainSet2 = modelCH[['gsin-category', 'end-user-entity-top']]
cluster2 = kmode.fit(trainSet2)
yPred2 = kmode.predict(trainSet2)
cluster2.cluster_centroids_
```

Out[ ]:
```
array([['R', 'Public Works and Government Services Canada'],
       ['T', 'Other'],
       ['5', 'Public Works and Government Services Canada']], dtype='<U43')
```

In [ ]:
```
plt.scatter(trainSet2['gsin-category'], trainSet2['end-user-entity-top'], s = 50, c = yPred2, cmap = plt.cm.Paired)
plt.show()
```



Creating Dummy Values

In [ ]:
```
#Dummy 1 for the dependent variable

cat1 = pd.get_dummies(modelCH['stantec-supplier'],drop_first=True)
cat1 = cat1.rename({'yes':'stantec-supplier-mod'}, axis = 1)
modelCH = pd.concat([modelCH,cat1],axis=1)
cat1.shape
```

Out[ ]:
```
(63394, 1)
```

In [ ]:
```
#Dummy 2 for gsin-description-top

cat2 = pd.get_dummies(modelCH['gsin-description-top'], prefix = 'top-gsin')
modelCH = pd.concat([modelCH,cat2],axis=1)
cat2.shape
```

Out[ ]:
```
(63394, 44)
```

In [ ]:
```
#Dummy 3 for gsin-category

cat3 = pd.get_dummies(modelCH['gsin-category'], prefix = 'gsin_cat')
modelCH = pd.concat([modelCH,cat3],axis=1)
cat3.shape
```

Out[ ]:
```
(63394, 11)
```

In [ ]:
```
#Dummy 4 for competitive-tender_en

cat4 = pd.get_dummies(modelCH['competitive-tender_en'], prefix = 'competitive_tender', drop_first=True)
modelCH = pd.concat([modelCH,cat4],axis=1)
cat4.shape
```

Out[ ]:
```
(63394, 1)
```

In [ ]:
```
#Dummy 5 for solicitation-procedure-description_en

cat5 = pd.get_dummies(modelCH['solicitation-procedure-description_en'], prefix = 'solicitation_procedure')
modelCH = pd.concat([modelCH,cat5],axis=1)
cat5.shape
```

Out[ ]:
```
(63394, 4)
```

In [ ]:
```
#Dummy 6 for trade-agreement

cat6 = pd.get_dummies(modelCH['trade-agreement'], prefix = 'trade-agreement')
modelCH = pd.concat([modelCH,cat6],axis=1)
cat6.shape
```

Out[ ]:
```
(63394, 33)
```

In [ ]:
```
#Dummy 7 for end-user-entity-top

cat7 = pd.get_dummies(modelCH['end-user-entity-top'], prefix = 'end_user')
modelCH = pd.concat([modelCH,cat7],axis=1)
cat7.shape
```

Out[ ]:
```
(63394, 20)
```

In [ ]:
```
#Dummy 8 for contracting-address-city-top

cat8 = pd.get_dummies(modelCH['contracting-address-city-top'], prefix = 'contracting-city')
modelCH = pd.concat([modelCH,cat8],axis=1)
cat8.shape
```

Out[ ]:
```
(63394, 19)
```

In [ ]:
```
#Dummy 9 for contracting-address-prov-state

cat9 = pd.get_dummies(modelCH['contracting-address-prov-state'], prefix = 'contracting_prov')
train = pd.concat([modelCH,cat9],axis=1)
cat9.shape
```

Out[ ]:
```
(63394, 13)
```

In [ ]:
```
#Dummy 10 for award month

cat10 = pd.get_dummies(modelCH['award month'], prefix = 'award month')
modelCH = pd.concat([modelCH,cat10],axis=1)
cat10.shape
```

Out[ ]:
```
(63394, 12)
```

In [ ]:
```
modelCH.columns

modelCH = modelCH.drop(['gsin-description-top', 'gsin-category', 'competitive-tender_en', 'contracting-address-postal-top', 'solicitation-procedure
```

Feature Evaluation

Manual Linear Regression Approach Using R Score

In [ ]:
```
#Creating a loop to enable model evaluation using r score on full set of variables - code development in progress

from sklearn import linear_model

featureEvalDatax = modelCH.drop(['stantec-supplier-mod', 'stantec-supplier'], axis = 1)
featureEvalDatay = modelCH['stantec-supplier-mod']

a=0
topAttribute = []
for attribute in featureEvalDatax:
    x = featureEvalDatax.iloc[:, a:a+1:1]
    y = featureEvalDatay
    regrFE = linear_model.LinearRegression().fit(x, y)
    topAttribute.append((regrFE.score(x, y), featureEvalDatax.columns[a]))
    a=a+1

len(topAttribute)
```

Out[ ]:
```
147
```

In [ ]:
```
topAttributeSort = sorted(topAttribute, key=lambda tup: tup[0], reverse = True)
print(topAttributeSort)
```

```
[(0.02786861309847022, 'gsin_cat_C'), (0.017317034250872387, 'top-gsin_Engineering Services - Buildings'), (0.00957377475996446, 'trade-agreement_
Z'), (0.006990385447206093, 'contracting-city_VANCOUVER'), (0.005368441613116781, 'end_user_Public Works and Government Services Canada'), (0.00397
0352469997218, 'top-gsin_Architect and Engineering Services - Buildings'), (0.0033554395959797256, 'gsin_cat_B'), (0.002869092648035809, 'trade-agr
eement_I'), (0.0021034368700646455, 'gsin_cat_T'), (0.002079336301876511, 'top-gsin_Architectual Services - Buildings'), (0.001824215293542486, 'gs
in_cat_R'), (0.0017054229343257399, 'trade-agreement_CKZ'), (0.0015617512604477746, 'top-gsin_Environmental Services'), (0.001547399542582872 4, 'en
d_user_Department of National Defence'), (0.0015375898359729634, 'gsin_cat_E'), (0.0012918849798727594, 'contracting-city_GATINEAU'), (0.0012871440
347090868, 'gsin_cat_5'), (0.0012092761727415802, 'trade-agreement_A7'), (0.0011506187452093863, 'gsin_cat_D'), (0.0011057815220405498, 'top-gsin_E
ngineering Services'), (0.001039258413460753, 'contracting-city_EDMONTON'), (0.0008962103828712431, 'trade-agreement_A3'), (0.0007602742903536353,
'top-gsin_Informatics Professional Services'), (0.0007453914849774312, 'days-since-first-award-normalized'), (0.0007407413045833477, 'top-gsin_Arch
itectural & Engineering Services - Highways, Roads, Railways,Bridges and Dams'), (0.0007188106259700122, 'award-year-normalized'), (0.0006271036884
623049, 'trade-agreement_N'), (0.0006130682731807502, 'contracting-city_OTTAWA'), (0.0006111717367541791, 'end_user_Other'), (0.000593406057060597,
'contracting-city_TORONTO'), (0.0005427045273540543, 'gsin_cat_A'), (0.0005202924393776787, 'top-gsin_Advertising Media Planning/Buying'), (0.00049
23766381257355, 'contracting-city_CALGARY'), (0.0003999954239691794, 'top-gsin_Human Resource Services, Business Consulting/Change Management; Proj
ect Management Services'), (0.00030430003818415763, 'contracting-city_WINNIPEG'), (0.0002676500871295895, 'top-gsin_Architect/Engineer Services - A
dministration & Service Buildings'), (0.00026492535912781, 'trade-agreement_Y'), (0.0002553784887324717, 'gsin_cat_U'), (0.0002467199868997 3496, 'e
nd_user_Parks Canada'), (0.0002418010868971976, 'top-gsin_Translation Services'), (0.0002368105315214697, 'contracting-city_HALIFAX'), (0.000231524
70610798837, 'top-gsin_Marine Architect and Engineering Services'), (0.00021679221489778744, 'top-gsin_Temporary Help Services, General Office Supp
ort'), (0.0001936512529202572, 'top-gsin_Information Products'), (0.0001874339235106 1594, 'contracting-city_Other'), (0.0001810890085177 6522, 'trad
e-agreement_X'), (0.00015919820016052633, 'top-gsin_Construction of Other Buildings'), (0.00015638691378994096, 'top-gsin_Audio Visual Production S
ervices'), (0.0001411723509651 1305, 'top-gsin_Military (R&D)'), (0.0001406017705249 9512, 'end_user_Health Canada'), (0.0001360920706764 4125, 'contr
acting-city_ST JOHNS'), (0.0001336864377590 1745, 'award month_10'), (0.0001328087808607492, 'award month_04'), (0.0001306564498025 9083, 'end_user_E
mployment and Social Development Canada'), (0.0001267565395786 0966, 'end_user_Canadian Space Agency'), (0.0001198646364887 2014, 'contracting-city_M
ISSISSAUGA'), (0.0001177691112789 2031, 'top-gsin_Product/Material - Design, Development, Formulation, Modification: Science and Technology Related
(R&D)'), (0.0001175664987053 1266, 'trade-agreement_DR'), (0.0001093963754701 3881, 'contracting-city_ST HUBERT'), (0.0001071375604925 2066, 'competit
ive_tender_Yes'), (0.0001071375604925 2066, 'solicitation_procedure_Traditional Non-Competitive'), (0.0001056633774527 6932, 'end_user_Agriculture an
d Agri-Food Canada'), (0.0001007403774569 9815, 'contracting-city_BORDEN'), (9.865339695658282e-05, 'end_user_Citizenship and Immigration Canada'),
(9.664134175968453e-05, 'solicitation_procedure_Open Bidding'), (9.448222954877572e-05, 'top-gsin_Advertising Related Services'), (8.974878528089647e-05, 'top-gsin_Miscellaneous Business Services'), (8.91247663890038e-05, 'top-gsin_Media Monitoring Services'), (8.629969242934799e-05, 'top-gsin
_Video Production Services'), (8.55565359721977e-05, 'top-gsin_Public Opinion Research - Quantitative'), (8.264710804239961e-05, 'top-gsin_Waterway
s, Harbours, Dams and Other Water Works'), (8.137428199961327e-05, 'end_user_Natural Resources Canada'), (8.020930006691351e-05, 'end_user_Foreign
Affairs, Trade And Development (Department Of)'), (7.8427309001094 51e-05, 'trade-agreement_A1'), (7.431484855979775e-05, 'end_user_Industry Canad
a'), (7.368210269576014e-05, 'top-gsin_Communication Promotional Material (including printing and identification)'), (7.29409535562553e-05, 'top-gs
in_General Marine Construction Work'), (7.07182129697781e-05, 'end_user_Veterans Affairs Canada'), (6.95331846545244e-05, 'top-gsin_Astronautics (R
&D)'), (6.77562070815485e-05, 'top-gsin_Electrical Installations and Major Repairs'), (6.568392288253122e-05, 'top-gsin_Press Release Distributio
```

n'), (6.553593785452172e-05, 'end_user_Royal Canadian Mounted Police'), (6.19648721240651e-05, 'solicitation_procedure_ACAN'), (5.843818163964176e-05, 'end_user_Aboriginal Affairs & Northern Development Canada'), (5.755172256238783e-05, 'top-gsin_Public Opinion Research - Quantitative & Qualitative'), (5.50410051800565e-05, 'top-gsin_Technical / Vocational Training'), (5.31219586313858e-05, 'top-gsin_Publications'), (5.2826790957527514e-05, 'top-gsin_Information Retrieval Services, Database'), (5.274180961400976e-05, 'award month_07'), (5.060167872339427e-05, 'top-gsin_Other'), (5.033095656259423e-05, 'top-gsin_Business Services'), (4.958118234088538e-05, 'top-gsin_General Office Help'), (4.958118234088538e-05, 'top-gsin_Highways, Roads, Railways, Airfield Runways'), (4.9138774811385844e-05, 'top-gsin_Public Opinion Research - Qualitative'), (4.825408602493475e-05, 'top-gsin_Language Interpretation Services'), (4.766438701564457e-05, 'top-gsin_Advertising Creative Services'), (4.7369565560506643e-05, 'top-gsin_General Contractor Services, Not Elsewhere Specified'), (4.6779978739808215e-05, 'trade-agreement_SB'), (4.545315433690522e-05, 'contracting-city_DARTMOUTH'), (4.0752628213902575e-05, 'award month_06'), (3.878177466365429e-05, 'end_user_Canada Revenue Agency'), (3.864262876063229e-05, 'end_user_Correctional Service of Canada'), (3.778532820852831e-05, 'award month_05'), (3.776868445648507e-05, 'contracting-city_MONTREAL'), (3.196385711889966e-05, 'end_user_Transport Canada'), (3.147075196729965e-05, 'top-gsin_Heating, Ventilation and Air Conditioning Maintenance & Inspection Services'), (2.9869920580294362e-05, 'trade-agreement_00'), (2.7536045466125536e-05, 'top-gsin_Internal and External Audits (Supply Arrangement PASS)'), (2.7121493492376914e-05, 'award month_08'), (2.4224215964041562e-05, 'trade-agreement_A4'), (2.2995974217066006e-05, 'top-gsin_End-to-End Learning Services (Excludes COTS Training)'), (1.9638461483451053e-05, 'trade-agreement_C'), (1.9440082405841608e-05, 'contracting-city_QUEBEC'), (1.8982441068127187e-05, 'award month_12'), (1.8877314651999377e-05, 'end_user_Environment Canada'), (1.8865239622800622e-05, 'contracting-city_Gatineau'), (1.7551644366120556e-05, 'trade-agreement_A'), (1.682845284933787e-05, 'trade-agreement_J'), (1.5414904294575038e-05, 'award month_11'), (1.5139047781054238e-05, 'trade-agreement_A6'), (1.5052958466155175e-05, 'trade-agreement_Q'), (1.1110378420053912e-05, 'contracting-city_CHARLOTTETOWN'), (9.047598969913473e-06, 'total-contract-value-normalized'), (7.321000304871816e-06, 'contracting-city_MONCTON'), (6.329035608043654e-06, 'award month_01'), (5.986090353404805e-06, 'trade-agreement_SA'), (5.436278018966512e-06, 'gsin_cat_S'), (5.109593214758235e-06, 'trade-agreement_A9'), (3.834152478954245e-06, 'award month_09'), (3.795158723507441 3e-06, 'trade-agreement_SC'), (3.788710569385678e-06, 'award month_03'), (2.6270859165844485e-06, 'trade-agreement_CKY'), (2.33511379044149e-06, 'trade-agreement_A5'), (1.4593079664670938e-06, 'trade-agreement_B'), (1.0214672296893212e-06, 'trade-agreement_H'), (1.011271469675279e-06, 'award month_02'), (9.822801640968493e-07, 'end_user_Fisheries and Oceans Canada'), (8.755295273221719e-07, 'trade-agreement_A8'), (8.755295273221719e-07, 'trade-agreement_G'), (7.910776491648619e-07, 'gsin_cat_H'), (3.5500281725386884e-07, 'trade-agreement_R'), (2.918247605787627e-07, 'trade-agreement_A2'), (1.459100784639844e-07, 'trade-agreement_CKI'), (1.459100784639844e-07, 'trade-agreement_CKX'), (1.459100784639844e-07, 'trade-agreement_O'), (7.130027313007048e-08, 'solicitation_procedure_Traditional Competitive'), (1.0104032055480161e-09, 'trade-agreement_W')]

Automated SelectKBest Approach Using F-Regression & F-Classif

In [ ]:
```
#Univariate Selection

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif

bestfeatures = SelectKBest(score_func = f_classif, k=30)
fitBF = bestfeatures.fit(featureEvalDatax,featureEvalDatay)
BFscores = pd.DataFrame(fitBF.scores_)
BFcolumns = pd.DataFrame(featureEvalDatax.columns)
featureScores = pd.concat([BFcolumns,BFscores],axis=1)
featureScores.columns = ['Specs','Score']
print(featureScores.nlargest(30,'Score'))
```

	Specs	Score
50	gsin_cat_C	1817.292544
18	top-gsin_Engineering Services - Buildings	1117.106405
95	trade-agreement_Z	612.767225
133	contracting-city_VANCOUVER	446.254002
112	end_user_Public Works and Government Services ...	342.153080
6	top-gsin_Architect and Engineering Services - ...	252.691860
49	gsin_cat_B	213.424159
83	trade-agreement_I	182.400846
56	gsin_cat_T	133.622136
8	top-gsin_Architectual Services - Buildings	132.087942
54	gsin_cat_R	115.851995
79	trade-agreement_CKZ	108.294859
19	top-gsin_Environmental Services	99.157395
102	end_user_Department of National Defence	98.244776
52	gsin_cat_E	97.620996
121	contracting-city_GATINEAU	82.001109
47	gsin_cat_5	81.699794
71	trade-agreement_A7	76.751249
51	gsin_cat_D	73.024046
17	top-gsin_Engineering Services	70.175301
120	contracting-city_EDMONTON	65.949208
67	trade-agreement_A3	56.863530
26	top-gsin_Informatics Professional Services	48.231977
1	days-since-first-award-normalized	47.287104
9	top-gsin_Architectural & Engineering Services ...	46.991882
2	award-year-normalized	45.599621
85	trade-agreement_N	39.778302
127	contracting-city_OTTAWA	38.887465
110	end_user_Other	38.767092
132	contracting-city_TORONTO	37.639532

Creating the Test & Training Set

In [ ]:
```
from sklearn.model_selection import KFold

kf5 = KFold(n_splits=5, shuffle=True, random_state = 2)
result = next(kf5.split(modelCH, None))
train = modelCH.iloc[result[0]]
test = modelCH.iloc[result[1]]
```

Unbalanced Multiple Linear Model with Original Top 12 Features

In [ ]:
```
xTrainEval = pd.DataFrame(train, columns = ['gsin_cat_C', 'top-gsin_Engineering Services - Buildings', 'trade-agreement_Z', 'contracting-city_VANCO
yTrainEval = pd.DataFrame(train, columns = ['stantec-supplier-mod'])
```

```
xTestEval = pd.DataFrame(test, columns = ['gsin_cat_C', 'top-gsin_Engineering Services - Buildings', 'trade-agreement_Z', 'contracting-city_VANCOUV
yTestEval = pd.DataFrame(test, columns = ['stantec-supplier-mod'])
```

In [ ]:
```
regr2 = linear_model.LinearRegression().fit(xTrainEval, yTrainEval)
predTrain = regr2.predict(xTrainEval)
predTest = regr2.predict(xTestEval)
#confusion_matrix(yTestEval, predTest)
yTestEval.value_counts()
#note that a confusion matrix does not work here because of the result is a probablility not an outcome - will need to swtich to a classification m
```

Out[ ]:
```
stantec-supplier-mod
0                      12552
1                        127
dtype: int64
```

In [ ]:
```
#0.01 used for prediction since approximately 127 of 12679 or 0.01% of the test values are positive
from sklearn.metrics import confusion_matrix

outcome = []
for pred in predTest:
    if (pred > 0.01):
        outcome.append(1)
    elif (pred <= 0.01):
        outcome.append(0)

confusion_matrix(yTestEval, outcome)
```

Out[ ]:
```
array([[10285,  2267],
       [   12,   115]], dtype=int64)
```

In [ ]:
```
from sklearn.metrics import classification_report

print(classification_report(yTestEval, outcome))
```

```
              precision    recall  f1-score   support

           0       1.00      0.82      0.90     12552
           1       0.05      0.91      0.09       127

    accuracy                           0.82     12679
   macro avg       0.52      0.86      0.50     12679
weighted avg       0.99      0.82      0.89     12679
```

In [ ]:
```
from sklearn.metrics import roc_auc_score

auc = roc_auc_score(yTestEval, outcome)
print(auc)
```

```
0.8624515715411291
```

Creating the Balanced Dataset for Use in All Models

In [ ]:
```
#Train Balanced model
count_class_0, count_class_1 = train['stantec-supplier'].value_counts()
ch_class_0 = train[train['stantec-supplier'] == 'no']
ch_class_1 = train[train['stantec-supplier'] == 'yes']
ch_class_0_under = ch_class_0.sample(count_class_1)
trainUnder = pd.concat([ch_class_0_under, ch_class_1], axis=0)
print(trainUnder['stantec-supplier'].value_counts())
```

```
no     454
yes    454
Name: stantec-supplier, dtype: int64
```

Balanced Multiple Linear Model with Original Top 12 Features

In [ ]:
```
xTrainEvalF = pd.DataFrame(trainUnder, columns = ['gsin_cat_C', 'top-gsin_Engineering Services - Buildings', 'trade-agreement_Z', 'contracting-city
yTrainEvalF = pd.DataFrame(trainUnder, columns = ['stantec-supplier-mod'])
xTestEvalF = pd.DataFrame(test, columns = ['gsin_cat_C', 'top-gsin_Engineering Services - Buildings', 'trade-agreement_Z', 'contracting-city_VANCOU
yTestEvalF = pd.DataFrame(test, columns = ['stantec-supplier-mod'])
```

In [ ]:
```
regrBalance9 = linear_model.LinearRegression().fit(xTrainEvalF, yTrainEvalF)
predTrainBalance9 = regrBalance9.predict(xTrainEvalF)
predTestBalance9 = regrBalance9.predict(xTestEvalF)

outcome9 = []
for pred in predTestBalance9:
    if (pred > 0.5):
        outcome9.append(1)
    elif (pred <= 0.5):
        outcome9.append(0)

confusion_matrix(yTestEvalF, outcome9)
```

Out[ ]:
```
array([[10826,  1726],
       [   16,   111]], dtype=int64)
```

In [ ]:
```
print(classification_report(yTestEvalF, outcome9))
```

```
              precision    recall  f1-score   support

           0       1.00      0.86      0.93     12552
           1       0.06      0.87      0.11       127

    accuracy                           0.86     12679
   macro avg       0.53      0.87      0.52     12679
weighted avg       0.99      0.86      0.92     12679
```

In [ ]:
```
print(roc_auc_score(yTestEvalF, outcome9))
```

```
0.8682538905868125
```

Balanced Multiple Linear Model with All Features

In [ ]:
```
trainUnderAll = trainUnder.drop(['stantec-supplier-mod', 'stantec-supplier'], axis = 1)
testAll = test.drop(['stantec-supplier-mod', 'stantec-supplier'], axis = 1)
```

In [ ]:
```
#Balanced regression model evaluation - All Features
xTrainEvalE = pd.DataFrame(trainUnderAll)
yTrainEvalE = pd.DataFrame(trainUnder, columns = ['stantec-supplier-mod'])
xTestEvalE = pd.DataFrame(testAll)
yTestEvalE = pd.DataFrame(test, columns = ['stantec-supplier-mod'])
```

In [ ]:
```
regrBalance4 = linear_model.LinearRegression().fit(xTrainEvalE, yTrainEvalE)
predTrainBalance4 = regrBalance4.predict(xTrainEvalE)
predTestBalance4 = regrBalance4.predict(xTestEvalE)

outcome4 = []
for pred in predTestBalance4:
    if (pred > 0.5):
        outcome4.append(1)
    elif (pred <= 0.5):
        outcome4.append(0)

confusion_matrix(yTestEvalE, outcome4)
```

Out[ ]:
```
array([[10531,  2021],
       [    9,   118]], dtype=int64)
```

In [ ]:
```
print(classification_report(yTestEvalE, outcome4))
```

```
              precision    recall  f1-score   support

           0       1.00      0.84      0.91     12552
           1       0.06      0.93      0.10       127

    accuracy                           0.84     12679
   macro avg       0.53      0.88      0.51     12679
weighted avg       0.99      0.84      0.90     12679
```

In [ ]:
```
auc = roc_auc_score(yTestEvalE, outcome4)
print(auc)
```

```
0.8840618303448207
```

Balanced Multiple Linear Model SelectKBest Top 16

In [ ]:
```
xTrainEvalB = pd.DataFrame(trainUnder, columns = ['gsin_cat_C', 'top-gsin_Engineering Services - Buildings', 'trade-agreement_Z', 'contracting-city
yTrainEvalB = pd.DataFrame(trainUnder, columns = ['stantec-supplier-mod'])
xTestEvalB = pd.DataFrame(test, columns = ['gsin_cat_C', 'top-gsin_Engineering Services - Buildings', 'trade-agreement_Z', 'contracting-city_VANCOU
yTestEvalB = pd.DataFrame(test, columns = ['stantec-supplier-mod'])
```

In [ ]:
```
regrBalance2 = linear_model.LinearRegression().fit(xTrainEvalB, yTrainEvalB)
predTrainBalance2 = regrBalance2.predict(xTrainEvalB)
predTestBalance2 = regrBalance2.predict(xTestEvalB)

outcome2 = []
for pred in predTestBalance2:
    if (pred > 0.5):
        outcome2.append(1)
    elif (pred <= 0.5):
        outcome2.append(0)

confusion_matrix(yTestEvalB, outcome2)
```

Out[ ]:
```
array([[10535,  2017],
       [   10,   117]], dtype=int64)
```

In [ ]:
```
print(classification_report(yTestEvalB, outcome2))
```

```
              precision    recall  f1-score   support

           0       1.00      0.84      0.91     12552
           1       0.05      0.92      0.10       127

    accuracy                           0.84     12679
   macro avg       0.53      0.88      0.51     12679
weighted avg       0.99      0.84      0.90     12679
```

In [ ]:
```
print(roc_auc_score(yTestEvalB, outcome2))
```

```
0.88028415962823
```

Logistic Regression Classifier

Using the same inputs as the Balanced regression model evaluation with SelectKBest Top 16 Features

In [ ]:
```
logReg = LogisticRegression()
```

In [ ]:
```
logRegFit2 = logReg.fit(xTrainEvalB, yTrainEvalB)
```

```
C:\Users\alana\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\utils\validation.py:985: DataConversionWarning: A column-vector y w
as passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [ ]:
```
logRegPred2 = logRegFit2.predict(xTestEvalB)
```

In [ ]:
```
score2 = logReg.score(xTestEvalB, yTestEvalB)
print(score2)
```

```
0.8574808738859532
```

In [ ]:
```
confMatrix2 = metrics.confusion_matrix(yTestEvalB, logRegPred2)
print(confMatrix2)
```

```
[[10757  1795]
 [   12   115]]
```

In [ ]:
```
print(classification_report(yTestEvalB, logRegPred2))
```

```
              precision    recall  f1-score   support

           0       1.00      0.86      0.92     12552
           1       0.06      0.91      0.11       127

    accuracy                           0.86     12679
   macro avg       0.53      0.88      0.52     12679
weighted avg       0.99      0.86      0.91     12679
```

In [ ]:
```
print(roc_auc_score(yTestEvalB, logRegPred2))
```

```
0.8812533561172922
```

Decision Trees Classifier

Using the same inputs as the Balanced regression model evaluation with SelectKBest Top 16 Features

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
```

In [ ]:
```
dtFit2 = dt.fit(xTrainEvalB, yTrainEvalB)
```

In [ ]:
```
dtPred2 = dtFit2.predict(xTestEvalB)
```

In [ ]:
```
dtScore2 = dtFit2.score(xTestEvalB, yTestEvalB)
print(dtScore2)
```

```
0.8608723085416831
```

In [ ]:
```
dtConfMatrix2 = metrics.confusion_matrix(yTestEvalB, dtPred2)
```

```
print(dtConfMatrix2)
```

```
[[10800  1752]
 [   12   115]]
```

In [ ]:
```
print(classification_report(yTestEvalB, dtPred2))
```

```
              precision    recall  f1-score   support

           0       1.00      0.86      0.92     12552
           1       0.06      0.91      0.12       127

    accuracy                           0.86     12679
   macro avg       0.53      0.88      0.52     12679
weighted avg       0.99      0.86      0.92     12679
```

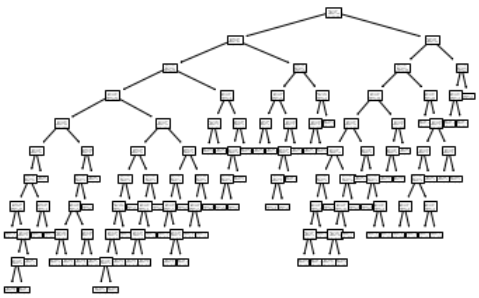
In [ ]:
```
print(roc_auc_score(yTestEvalB, dtPred2))
```

```
0.8829662305596121
```

In [ ]:
```
tree.plot_tree(dtFit2)
```

Out[ ]:
```
[Text(234.27516891891895, 207.55636363636364, 'X[0] <= 0.5\ngini = 0.5\nsamples = 908\nvalue = [454, 454]'),
 Text(164.00675675675677, 187.7890909090909, 'X[14] <= 0.5\ngini = 0.383\nsamples = 543\nvalue = [403, 140]'),
 Text(117.63243243243244, 168.0218181818182, 'X[6] <= 0.5\ngini = 0.313\nsamples = 490\nvalue = [395, 95]'),
 Text(76.91351351351352, 148.25454545454545, 'X[4] <= 0.5\ngini = 0.208\nsamples = 432\nvalue = [381, 51]'),
 Text(40.718918918918924, 128.48727272727274, 'X[7] <= 0.5\ngini = 0.074\nsamples = 208\nvalue = [200, 8]'),
 Text(22.621621621621625, 108.72, 'X[3] <= 0.5\ngini = 0.041\nsamples = 144\nvalue = [141, 3]'),
 Text(18.0972972972973, 88.95272727272729, 'X[10] <= 0.5\ngini = 0.028\nsamples = 142\nvalue = [140, 2]'),
 Text(9.04864864864865, 69.18545454545455, 'X[13] <= 0.5\ngini = 0.015\nsamples = 130\nvalue = [129, 1]'),
 Text(4.524324324324325, 49.418181818181836, 'gini = 0.0\nsamples = 87\nvalue = [87, 0]'),
 Text(13.572972972972973, 49.418181818181836, 'X[8] <= 0.5\ngini = 0.045\nsamples = 43\nvalue = [42, 1]'),
 Text(9.04864864864865, 29.650909090909096, 'X[2] <= 0.5\ngini = 0.067\nsamples = 29\nvalue = [28, 1]'),
 Text(4.524324324324325, 9.883636363636384, 'gini = 0.069\nsamples = 28\nvalue = [27, 1]'),
 Text(13.572972972972973, 9.883636363636384, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(18.0972972972973, 29.650909090909096, 'gini = 0.0\nsamples = 14\nvalue = [14, 0]'),
 Text(27.145945945945947, 69.18545454545455, 'X[13] <= 0.5\ngini = 0.153\nsamples = 12\nvalue = [11, 1]'),
 Text(22.621621621621625, 49.418181818181836, 'gini = 0.0\nsamples = 7\nvalue = [7, 0]'),
 Text(31.670270270270272, 49.418181818181836, 'gini = 0.32\nsamples = 5\nvalue = [4, 1]'),
 Text(27.145945945945947, 88.95272727272729, 'gini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(58.81621621621622, 108.72, 'X[13] <= 0.5\ngini = 0.144\nsamples = 64\nvalue = [59, 5]'),
 Text(54.29189189189189, 88.95272727272729, 'X[3] <= 0.5\ngini = 0.224\nsamples = 39\nvalue = [34, 5]'),
 Text(49.767567567567575, 69.18545454545455, 'X[8] <= 0.5\ngini = 0.245\nsamples = 35\nvalue = [30, 5]'),
 Text(40.718918918918924, 49.418181818181836, 'X[10] <= 0.5\ngini = 0.287\nsamples = 23\nvalue = [19, 4]'),
 Text(36.1945945945946, 29.650909090909096, 'gini = 0.278\nsamples = 18\nvalue = [15, 3]'),
 Text(45.24324324324325, 29.650909090909096, 'gini = 0.32\nsamples = 5\nvalue = [4, 1]'),
 Text(58.81621621621622, 49.418181818181836, 'X[15] <= 0.5\ngini = 0.153\nsamples = 12\nvalue = [11, 1]'),
 Text(54.29189189189189, 29.650909090909096, 'gini = 0.18\nsamples = 10\nvalue = [9, 1]'),
 Text(63.340540540540545, 29.650909090909096, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(58.81621621621622, 69.18545454545455, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(63.340540540540545, 88.95272727272729, 'gini = 0.0\nsamples = 25\nvalue = [25, 0]'),
 Text(113.10810810810811, 128.48727272727274, 'X[7] <= 0.5\ngini = 0.31\nsamples = 224\nvalue = [181, 43]'),
 Text(95.01081081081082, 108.72, 'X[10] <= 0.5\ngini = 0.385\nsamples = 142\nvalue = [105, 37]'),
 Text(85.96216216216217, 88.95272727272729, 'X[2] <= 0.5\ngini = 0.423\nsamples = 79\nvalue = [55, 24]'),
 Text(81.43783783783785, 69.18545454545455, 'X[3] <= 0.5\ngini = 0.429\nsamples = 77\nvalue = [53, 24]'),
 Text(76.91351351351352, 49.418181818181836, 'X[15] <= 0.5\ngini = 0.425\nsamples = 75\nvalue = [52, 23]'),
 Text(72.3891891891892, 29.650909090909096, 'X[8] <= 0.5\ngini = 0.433\nsamples = 63\nvalue = [43, 20]'),
 Text(67.86486486486487, 9.883636363636384, 'gini = 0.437\nsamples = 59\nvalue = [40, 19]'),
 Text(76.91351351351352, 9.883636363636384, 'gini = 0.375\nsamples = 4\nvalue = [3, 1]'),
 Text(81.43783783783785, 29.650909090909096, 'gini = 0.375\nsamples = 12\nvalue = [9, 3]'),
 Text(85.96216216216217, 49.418181818181836, 'gini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(90.4864864864865, 69.18545454545455, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(104.05945945945948, 88.95272727272729, 'X[2] <= 0.5\ngini = 0.328\nsamples = 63\nvalue = [50, 13]'),
 Text(99.53513513513515, 69.18545454545455, 'X[3] <= 0.5\ngini = 0.335\nsamples = 61\nvalue = [48, 13]'),
 Text(95.01081081081082, 49.418181818181836, 'X[15] <= 0.5\ngini = 0.328\nsamples = 58\nvalue = [46, 12]'),
 Text(90.4864864864865, 29.650909090909096, 'gini = 0.351\nsamples = 22\nvalue = [17, 5]'),
 Text(99.53513513513515, 29.650909090909096, 'gini = 0.313\nsamples = 36\nvalue = [29, 7]'),
 Text(104.05945945945948, 49.418181818181836, 'gini = 0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(108.58378378378379, 69.18545454545455, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(131.20540540540543, 108.72, 'X[10] <= 0.5\ngini = 0.136\nsamples = 82\nvalue = [76, 6]'),
 Text(122.15675675675676, 88.95272727272729, 'X[3] <= 0.5\ngini = 0.171\nsamples = 53\nvalue = [48, 5]'),
 Text(117.63243243243244, 69.18545454545455, 'X[8] <= 0.5\ngini = 0.177\nsamples = 51\nvalue = [46, 5]'),
 Text(113.10810810810811, 49.418181818181836, 'gini = 0.198\nsamples = 36\nvalue = [32, 4]'),
 Text(122.15675675675676, 49.418181818181836, 'X[15] <= 0.5\ngini = 0.124\nsamples = 15\nvalue = [14, 1]'),
 Text(117.63243243243244, 29.650909090909096, 'gini = 0.133\nsamples = 14\nvalue = [13, 1]'),
 Text(126.68108108108109, 29.650909090909096, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(126.68108108108109, 69.18545454545455, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(140.25405405405408, 88.95272727272729, 'X[15] <= 0.5\ngini = 0.067\nsamples = 29\nvalue = [28, 1]'),
 Text(135.72972972972974, 69.18545454545455, 'X[3] <= 0.5\ngini = 0.08\nsamples = 24\nvalue = [23, 1]'),
 Text(131.20540540540543, 49.418181818181836, 'gini = 0.083\nsamples = 23\nvalue = [22, 1]'),
 Text(140.25405405405408, 49.418181818181836, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(144.7783783783784, 69.18545454545455, 'gini = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(158.35135135135135, 148.25454545454545, 'X[4] <= 0.5\ngini = 0.366\nsamples = 58\nvalue = [14, 44]'),
 Text(149.3027027027027, 128.48727272727274, 'X[7] <= 0.5\ngini = 0.32\nsamples = 5\nvalue = [4, 1]'),
 Text(144.7783783783784, 108.72, 'gini = 0.444\nsamples = 3\nvalue = [2, 1]'),
 Text(153.82702702702704, 108.72, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(167.4, 128.48727272727274, 'X[15] <= 0.5\ngini = 0.306\nsamples = 53\nvalue = [10, 43]'),
 Text(162.8756756756757, 108.72, 'X[7] <= 0.5\ngini = 0.249\nsamples = 48\nvalue = [7, 41]'),
 Text(158.35135135135135, 88.95272727272729, 'X[2] <= 0.5\ngini = 0.231\nsamples = 45\nvalue = [6, 39]'),
```

```
Text(153.82702702702704, 69.18545454545455, 'gini = 0.21\nsamples = 42\nvalue = [5, 37]'),
Text(162.8756756756757, 69.18545454545455, 'gini = 0.444\nsamples = 3\nvalue = [1, 2]'),
Text(167.4, 88.95272727272729, 'gini = 0.444\nsamples = 3\nvalue = [1, 2]'),
Text(171.92432432432435, 108.72, 'gini = 0.48\nsamples = 5\nvalue = [3, 2]'),
Text(210.3810810810811, 168.0218181818182, 'X[12] <= 0.5\ngini = 0.256\nsamples = 53\nvalue = [8, 45]'),
Text(194.54594594594596, 148.25454545454545, 'X[4] <= 0.5\ngini = 0.34\nsamples = 23\nvalue = [5, 18]'),
Text(185.4972972972973, 128.48727272727274, 'X[13] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
Text(180.972972972973, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(190.02162162162165, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(203.5945945945946, 128.48727272727274, 'X[2] <= 0.5\ngini = 0.308\nsamples = 21\nvalue = [4, 17]'),
Text(199.0702702702703, 108.72, 'X[15] <= 0.5\ngini = 0.32\nsamples = 20\nvalue = [4, 16]'),
Text(194.54594594594596, 88.95272727272729, 'X[7] <= 0.5\ngini = 0.332\nsamples = 19\nvalue = [4, 15]'),
Text(190.02162162162165, 69.18545454545455, 'gini = 0.346\nsamples = 18\nvalue = [4, 14]'),
Text(199.0702702702703, 69.18545454545455, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(203.5945945945946, 88.95272727272729, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(208.11891891891895, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(226.21621621621622, 148.25454545454545, 'X[3] <= 0.5\ngini = 0.18\nsamples = 30\nvalue = [3, 27]'),
Text(221.6918918918919, 128.48727272727274, 'X[4] <= 0.5\ngini = 0.142\nsamples = 26\nvalue = [2, 24]'),
Text(217.16756756756757, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(226.21621621621622, 108.72, 'gini = 0.147\nsamples = 25\nvalue = [2, 23]'),
Text(230.74054054054056, 128.48727272727274, 'gini = 0.375\nsamples = 4\nvalue = [1, 3]'),
Text(304.54358108108113, 187.7890909090909, 'X[13] <= 0.5\ngini = 0.24\nsamples = 365\nvalue = [51, 314]'),
Text(283.3358108108108, 168.0218181818182, 'X[2] <= 0.5\ngini = 0.224\nsamples = 358\nvalue = [46, 312]'),
Text(263.5418918918919, 148.25454545454545, 'X[9] <= 0.5\ngini = 0.262\nsamples = 258\nvalue = [40, 218]'),
Text(246.5756756756757, 128.48727272727274, 'X[3] <= 0.5\ngini = 0.237\nsamples = 226\nvalue = [31, 195]'),
Text(235.26486486486488, 108.72, 'X[4] <= 0.5\ngini = 0.261\nsamples = 175\nvalue = [27, 148]'),
Text(226.21621621621622, 88.95272727272729, 'X[1] <= 0.5\ngini = 0.469\nsamples = 8\nvalue = [3, 5]'),
Text(221.6918918918919, 69.18545454545455, 'X[7] <= 0.5\ngini = 0.48\nsamples = 5\nvalue = [3, 2]'),
Text(217.16756756756757, 49.418181818181836, 'X[5] <= 0.5\ngini = 0.5\nsamples = 4\nvalue = [2, 2]'),
Text(212.64324324324326, 29.650909090909096, 'gini = 0.444\nsamples = 3\nvalue = [1, 2]'),
Text(221.6918918918919, 29.650909090909096, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(226.21621621621622, 49.418181818181836, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(230.74054054054056, 69.18545454545455, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]'),
Text(244.31351351351353, 88.95272727272729, 'X[7] <= 0.5\ngini = 0.246\nsamples = 167\nvalue = [24, 143]'),
Text(239.78918918918922, 69.18545454545455, 'X[1] <= 0.5\ngini = 0.251\nsamples = 163\nvalue = [24, 139]'),
Text(235.26486486486488, 49.418181818181836, 'X[5] <= 0.5\ngini = 0.24\nsamples = 136\nvalue = [19, 117]'),
Text(230.74054054054056, 29.650909090909096, 'gini = 0.243\nsamples = 127\nvalue = [18, 109]'),
Text(239.78918918918922, 29.650909090909096, 'gini = 0.198\nsamples = 9\nvalue = [1, 8]'),
Text(244.31351351351353, 49.418181818181836, 'gini = 0.302\nsamples = 27\nvalue = [5, 22]'),
Text(248.83783783783787, 69.18545454545455, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
Text(257.8864864864865, 108.72, 'X[1] <= 0.5\ngini = 0.145\nsamples = 51\nvalue = [4, 47]'),
Text(253.36216216216218, 88.95272727272729, 'gini = 0.0\nsamples = 10\nvalue = [0, 10]'),
Text(262.41081081081086, 88.95272727272729, 'X[4] <= 0.5\ngini = 0.176\nsamples = 41\nvalue = [4, 37]'),
Text(257.8864864864865, 69.18545454545455, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
Text(266.93513513513517, 69.18545454545455, 'X[7] <= 0.5\ngini = 0.193\nsamples = 37\nvalue = [4, 33]'),
Text(262.41081081081086, 49.418181818181836, 'gini = 0.202\nsamples = 35\nvalue = [4, 31]'),
Text(271.4594594594595, 49.418181818181836, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
Text(280.50810810810816, 128.48727272727274, 'X[7] <= 0.5\ngini = 0.404\nsamples = 32\nvalue = [9, 23]'),
Text(275.9837837837838, 108.72, 'X[3] <= 0.5\ngini = 0.412\nsamples = 31\nvalue = [9, 22]'),
Text(271.4594594594595, 88.95272727272729, 'gini = 0.413\nsamples = 24\nvalue = [7, 17]'),
Text(280.50810810810816, 88.95272727272729, 'gini = 0.408\nsamples = 7\nvalue = [2, 5]'),
Text(285.0324324324325, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(303.1297297297298, 148.25454545454545, 'X[4] <= 0.5\ngini = 0.113\nsamples = 100\nvalue = [6, 94]'),
Text(298.6054054054054, 128.48727272727274, 'gini = 0.5\nsamples = 2\nvalue = [1, 1]'),
Text(307.6540540540541, 128.48727272727274, 'X[1] <= 0.5\ngini = 0.097\nsamples = 98\nvalue = [5, 93]'),
Text(298.6054054054054, 108.72, 'X[5] <= 0.5\ngini = 0.126\nsamples = 59\nvalue = [4, 55]'),
Text(294.0810810810811, 88.95272727272729, 'X[3] <= 0.5\ngini = 0.159\nsamples = 46\nvalue = [4, 42]'),
Text(285.0324324324325, 69.18545454545455, 'X[9] <= 0.5\ngini = 0.191\nsamples = 28\nvalue = [3, 25]'),
Text(280.50810810810816, 49.418181818181836, 'gini = 0.159\nsamples = 23\nvalue = [2, 21]'),
Text(289.5567567567568, 49.418181818181836, 'gini = 0.32\nsamples = 5\nvalue = [1, 4]'),
Text(303.1297297297298, 69.18545454545455, 'X[9] <= 0.5\ngini = 0.105\nsamples = 18\nvalue = [1, 17]'),
Text(298.6054054054054, 49.418181818181836, 'gini = 0.245\nsamples = 7\nvalue = [1, 6]'),
Text(307.6540540540541, 49.418181818181836, 'gini = 0.0\nsamples = 11\nvalue = [0, 11]'),
Text(303.1297297297298, 88.95272727272729, 'gini = 0.0\nsamples = 13\nvalue = [0, 13]'),
Text(316.7027027027027, 108.72, 'X[3] <= 0.5\ngini = 0.05\nsamples = 39\nvalue = [1, 38]'),
Text(312.1783783783784, 88.95272727272729, 'gini = 0.0\nsamples = 9\nvalue = [0, 9]'),
Text(321.2270270270271, 88.95272727272729, 'gini = 0.064\nsamples = 30\nvalue = [1, 29]'),
Text(325.7513513513514, 168.0218181818182, 'X[7] <= 0.5\ngini = 0.408\nsamples = 7\nvalue = [5, 2]'),
Text(321.2270270270271, 148.25454545454545, 'X[5] <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
Text(316.7027027027027, 128.48727272727274, 'gini = 0.5\nsamples = 2\nvalue = [1, 1]'),
Text(325.7513513513514, 128.48727272727274, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(330.2756756756757, 148.25454545454545, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]')]
```

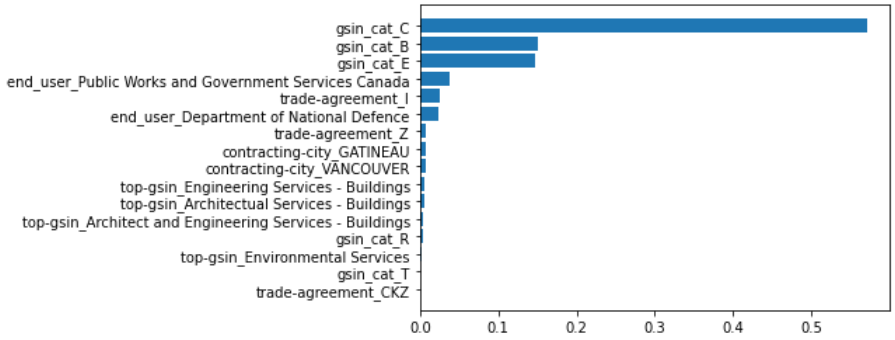


In [ ]:

```
sorted_idx2 = dt.feature_importances_.argsort()
plt.barh(xTrainEvalB.columns[sorted_idx2],dt.feature_importances_[sorted_idx2])
```

Out[ ]:

```
<BarContainer object of 16 artists>
```



Random Forest Classifier

Using the same inputs as the Balanced regression model evaluation with SelectKBest Top 16 Features

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
```

In [ ]:
```
rfFit2 = rf.fit(xTrainEvalB, yTrainEvalB)
```

```
C:\Users\alana\AppData\Local\Temp/ipykernel_1824/135279312.py:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  rfFit2 = rf.fit(xTrainEvalB, yTrainEvalB)
```

In [ ]:
```
rfPred2 = rfFit2.predict(xTestEvalB)
```

In [ ]:
```
rfScore2 = rfFit2.score(xTestEvalB, yTestEvalB)
print(rfScore2)
```

```
0.8563766858585062
```

In [ ]:
```
rfConfMatrix = metrics.confusion_matrix(yTestEvalB, rfPred2)
print(rfConfMatrix)
```

```
[[10741  1811]
 [   10   117]]
```

In [ ]:
```
print(classification_report(yTestEvalB, rfPred2))
```

```
              precision    recall  f1-score   support

           0       1.00      0.86      0.92     12552
           1       0.06      0.92      0.11       127

    accuracy                           0.86     12679
   macro avg       0.53      0.89      0.52     12679
weighted avg       0.99      0.86      0.91     12679
```

In [ ]:
```
print(roc_auc_score(yTestEvalB, rfPred2))
```

```
0.8884900232356232
```

In [ ]:
```
sorted_idx3 = rf.feature_importances_.argsort()
plt.barh(xTrainEvalB.columns[sorted_idx3],rf.feature_importances_[sorted_idx3])
plt.xlabel('Number of Instances')
```

Out[ ]:
```
Text(0.5, 0, 'Number of Instances')
```