Classification Project

Problem Statement

- I am going classify the different types of customer complaints that were made to Santander bank in order see if there is a relationships between the type of complaint, the complaint outcome, and if the customer disputes the outcome.

Overveiw

- I am going to look at the data set and see if any conclusions can be drawn from the data by completing different classification groups. I going to see if there are any relationships between the types of complaints and the sub-issues and depending on how successful this is I would like to try and link the complaints back to the different companies.

Sourcing the data

The data is freely available in the public domain and can be found on the Kaggle website free of charge. (URL: https://www.kaggle.com/selener/consumer-complaint-database (https://www.kaggle.com/selener/consumer-complaint-database))

The data is non-continuous therefore suitable for classification analysis. This file does not contain any sensitive information as it is in the public domain.

This data can be linked to my role in UBS and the wider organisation as the bank will get similar client complaints. There are a numerous different departments and products offered by UBS and there are hundreds of different clients serviced. Providing excellent customer service is a goal for UBS and this achieved most of the time it would unrealistic of me not to acknowledge it would be virtually impossible to keep the entire customer base happy. Clients do complain when there have been issues.

You can take the methodology used in this model and apply it to internal complaints data. Once this has been completed you can analysis the most common client complaints and focus resources on trying to improve in these areas. This project is also going to be used an example of the power the classification algorithms and can be used as a starting point for discussions on where they can be implemented within UBS.

Within my role as Offboarding Analytics and Triage lead, I can use these algorithms to classify where requests have come from and who is submitting the most requests. The team could also start recording the times when stakeholders have had issues with our process and build out a similar model to help report to senior management the areas we can improve on. I am attempting to demonstrate the business case for this by using data which is not niche to demonstrate their flexibility.

Tasks

I will need to do the following tasks to successfully complete this task:

In order for this analysis to be performed to a useful level I will have to:

1. Import, clean and check the data types.
    - The data I am using is taken from a public source. I predict there might be a number of data quality issues due to nature of where this data is sourced from. In order to make it usable and create better analysis I will need to perform a data clean-up.
    - This will involve making sure numbers are classed as integers to make sure the analysis can be performed correctly

1. Data Visualisation.
    - Visualising the data in different ways should give valuable insight into any potential relationships in the data. It will also highlight any potential outliers in the population and data quality issues that might have been missed in the original clean-up exercise.
    - Bar charts immediately strike me as the most useful for this exercise.

1. Correlation analysis
    - I am going to perform some correlation analysis to get insight into relationships within the data. There might be relationships in the data which are not obvious from the initial visualisations.

1. Statistical analysis
    - I am going to try and work out the r-coefficients and p-values for the data as this can provide further insights into the data set.

1. Classification models

    - I'm going to use two different classification models to see which is the most suitable and accurate for classifying the data.
        A. Random Forest
        B. Decision Tree These are common models which use different techniques for their analysis. I will do into more detail on these later on.
2. Conclusion

    - Has my clustering been successful, what can this analysis tell me, and how can I use it to improve the process I manage?

I have completed this analysis on my own without any help from a team.

Importing the libraries

In [1]:

```
# importing panads and numpy for data/string manipulation
import pandas as pd
import numpy as np

# importing re in case I need to use regular expressions to manipulate string data
import re

# importing pyplot from matplotlib and seaborn for visualiation
import matplotlib.pyplot as plt
import seaborn as sns

#importing sklearn to test my model
from sklearn import tree, metrics
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# importing warning to repress the warnings regarding future versions of this software
import warnings
warnings.simplefilter(action = 'ignore', category = FutureWarning)
```

Importing and exploring the data

In [2]:

```
# Load the data
data = pd.read_csv("rows.csv")
```

```
C:\SnapVolumesTemp\MountPoints\{8f481e4a-1b0a-11ea-93e2-0050568ce416}\SVRO
OT\UBS\Dev\Miniconda\lib\site-packages\IPython\core\interactiveshell.py:30
18: DtypeWarning: Columns (4,5,6,11,16) have mixed types. Specify dtype op
tion on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

I should be importing this data set with the low_memory set as false so it all imports correctly.

In [3]:

```
# Load the data
data = pd.read_csv("rows.csv", low_memory = False)
```

In [4]:

```
# print out the first 5 rows to make sure the data has been imported correctly
data.head()
```

Out[4]:

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Com
0	05/10/2019	Checking or savings account	Checking account	Managing an account	Problem using a debit or ATM card	NaN	NaN	N FEDE CR UI
1	05/10/2019	Checking or savings account	Other banking product or service	Managing an account	Deposits and withdrawals	NaN	NaN	BOI EMPLOY CR UI
2	05/10/2019	Debt collection	Payday loan debt	Communication tactics	Frequent or repeated calls	NaN	NaN	C Interme Hol
3	05/10/2019	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Old information reappears or never goes away	NaN	NaN	Ad Rec Service
4	05/10/2019	Checking or savings account	Checking account	Managing an account	Banking errors	NaN	NaN	FINAN

There looks like there are a lot of null values in this data. These might be just noise in the data. I will confirm these as part of my data clean up.

I will potentially drop these null values when I clean the data for the first model.

In [5]:

```
#describing the data to see which columns have complete data
data["Complaint ID"].describe()
```

Out[5]:

```
count    1.282355e+06
mean     1.929831e+06
std      9.645326e+05
min      1.000000e+00
25%      1.124152e+06
50%      2.123218e+06
75%      2.798886e+06
max      3.238682e+06
Name: Complaint ID, dtype: float64
```

In [6]:

```
#check the data shape
data.shape
```

Out[6]:

```
(1282355, 18)
```

As I have a very large dataset I will strip some of the unnecessary columns out to make the data cleaner.

Cleaning the data

In [7]:

```
# show all column headers
list(data.columns.values)
```

Out[7]:

```
['Date received',
 'Product',
 'Sub-product',
 'Issue',
 'Sub-issue',
 'Consumer complaint narrative',
 'Company public response',
 'Company',
 'State',
 'ZIP code',
 'Tags',
 'Consumer consent provided?',
 'Submitted via',
 'Date sent to company',
 'Company response to consumer',
 'Timely response?',
 'Consumer disputed?',
 'Complaint ID']
```

In [8]:

```
data.shape
```

Out[8]:

```
(1282355, 18)
```

There are over one million rows in this data. I am dealing with a large amount of data

In [9]:

```
# drop any rows missing data (null/NaN)
data = data.dropna(how = 'any')
```

In [10]:

```
# check the impact this has had on the overall data.
data.shape
```

Out[10]:

```
(2301, 18)
```

This acton has dramatically reduced my dataset. I will use this reduced data set for my initial model and this smaller data population makes a good pilot run for the model.

Going back to the full dataset once the model is completed would be a good second test.

In [11]:

```
# check the data types
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2301 entries, 514012 to 912429
Data columns (total 18 columns):
Date received                   2301 non-null object
Product                         2301 non-null object
Sub-product                     2301 non-null object
Issue                           2301 non-null object
Sub-issue                       2301 non-null object
Consumer complaint narrative    2301 non-null object
Company public response         2301 non-null object
Company                         2301 non-null object
State                           2301 non-null object
ZIP code                        2301 non-null object
Tags                            2301 non-null object
Consumer consent provided?      2301 non-null object
Submitted via                   2301 non-null object
Date sent to company            2301 non-null object
Company response to consumer    2301 non-null object
Timely response?                2301 non-null object
Consumer disputed?              2301 non-null object
Complaint ID                    2301 non-null int64
dtypes: int64(1), object(17)
memory usage: 341.6+ KB
```

```
In [12]:
```

```
# removing the punctuation from column headers to make them easier to work with
data.rename(columns={'Timely response?':'Timely response',
                     'Consumer disputed?':'Consumer disputed'}, inplace = True)
```

For my first test I have decided to strip out everything but the complaint specific data to see if I can classify the products with the most issues. I can then attempt to include the companies to see what companies have the most type of complaints against particular products.

```
In [13]:
```

```
# create new variable
complaints = data[['Product','Sub-product','Issue','Sub-issue','Complaint ID','Timely r
esponse','Consumer disputed']]
```

```
In [14]:
```

```
# check this has been successful
complaints.head()
```

```
Out[14]:
```

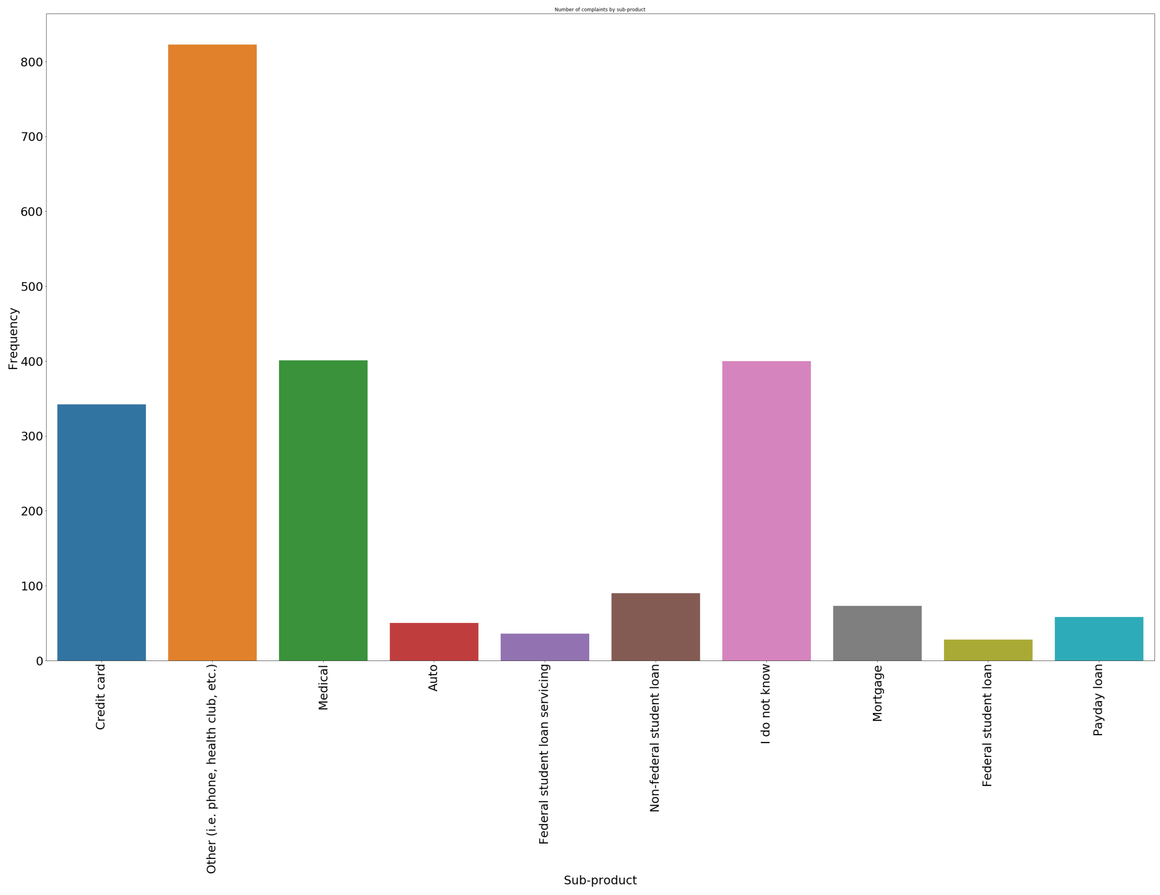
	Product	Sub-product	Issue	Sub-issue	Complaint ID	Timely response	Consumer disputed
514012	Debt collection	Credit card	False statements or representation	Attempted to collect wrong amount	2446820	Yes	No
514353	Debt collection	Credit card	Taking/threatening an illegal action	Attempted to/Collected exempt funds	2447164	Yes	No
515232	Debt collection	Other (i.e. phone, health club, etc.)	False statements or representation	Attempted to collect wrong amount	2445095	Yes	No
515509	Debt collection	Medical	Cont'd attempts collect debt not owed	Debt is not mine	2442145	Yes	No
515702	Debt collection	Medical	Cont'd attempts collect debt not owed	Debt was paid	2441597	Yes	No

Visualisations of the data

There are number of different companies in this data so I am visualising them out to help me make a decision on what to look into.

In [15]:

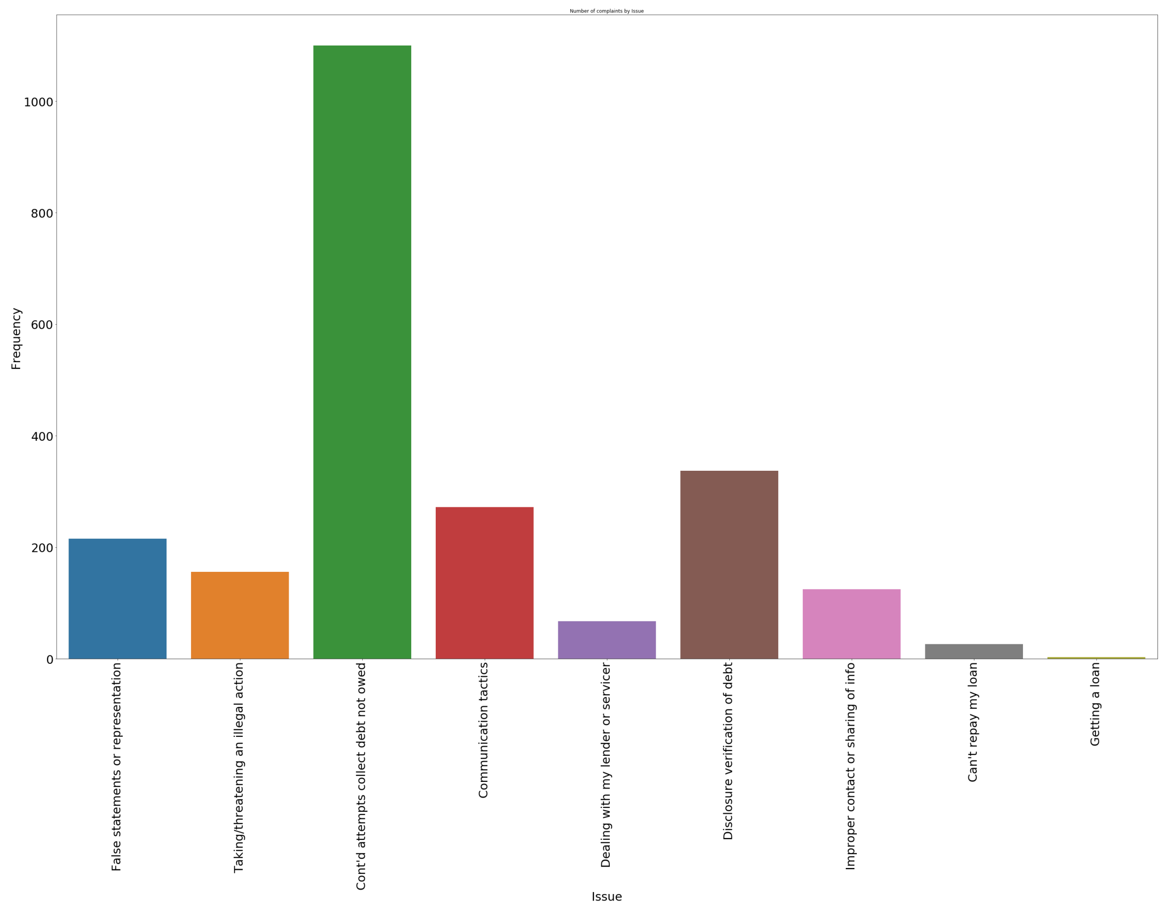
```
fig3 = sns.countplot(x='Sub-product', data = complaints)
fig3.figure.set_size_inches(50,30)
fig3.set_xlabel("Sub-product", fontsize=30)
fig3.set_ylabel("Frequency", fontsize=30)
plt.title("Number of complaints by sub-product")
plt.xticks(fontsize = 14, rotation=90)
fig3.tick_params(labelsize = 30)
plt.show()
```



From the above I can see that the most complaints are about "Other" products. Apart from the "I do not know category" it is 'Medical' followed by 'Credit Cards'. Having an "other" category in the sub-product reduces the potential accuracy is I use this category for the model. I am now going to plot out the Issues and Sub-issue.

In [16]:

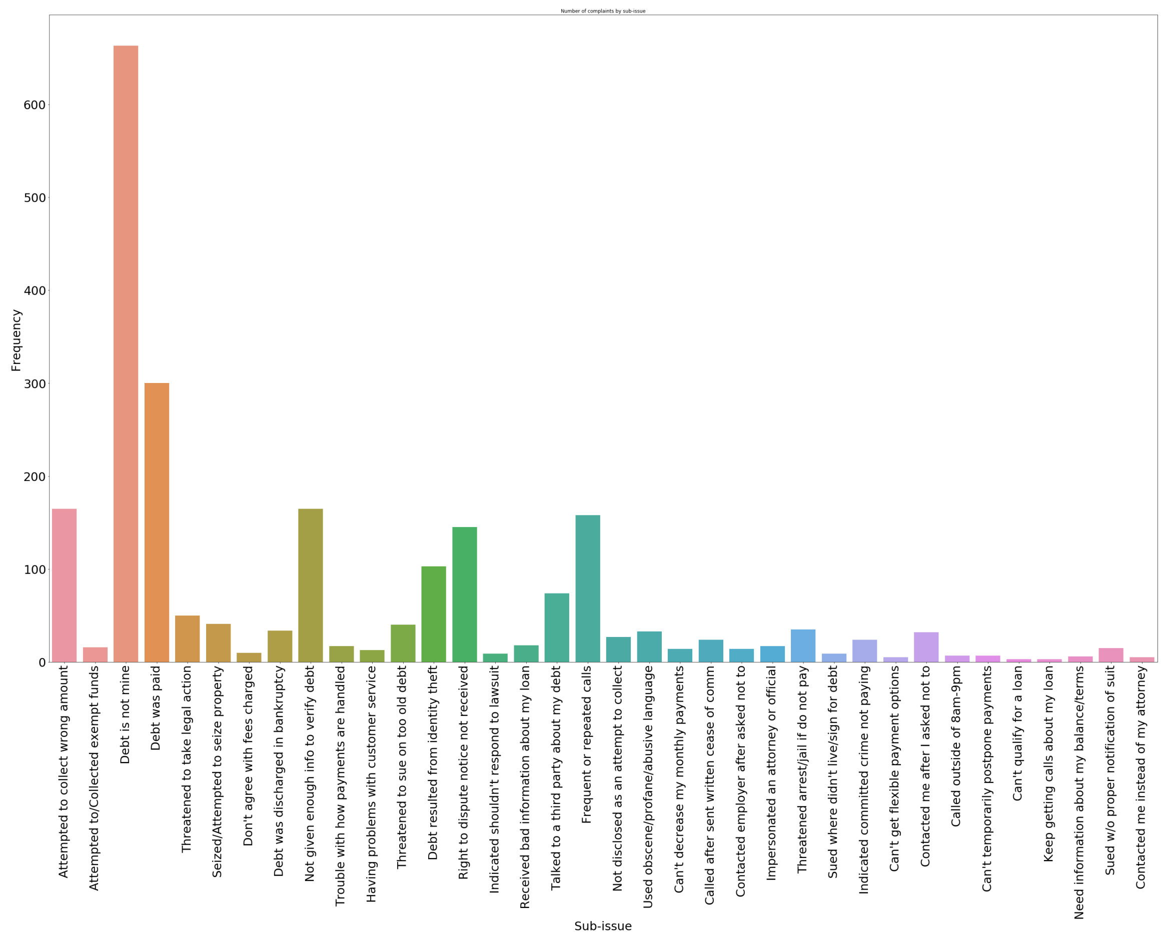
```
fig4 = sns.countplot(x='Issue', data = complaints)
fig4.figure.set_size_inches(50,30)
fig4.set_xlabel("Issue", fontsize=30)
fig4.set_ylabel("Frequency", fontsize=30)
plt.title("Number of complaints by Issue")
plt.xticks(rotation=90)
fig4.tick_params(labelsize = 30)
plt.show()
```



It makes sense customers complained the most about attempts to collect debt not owned. If there is no debt than the bank is clearly in the wrong.

In [17]:

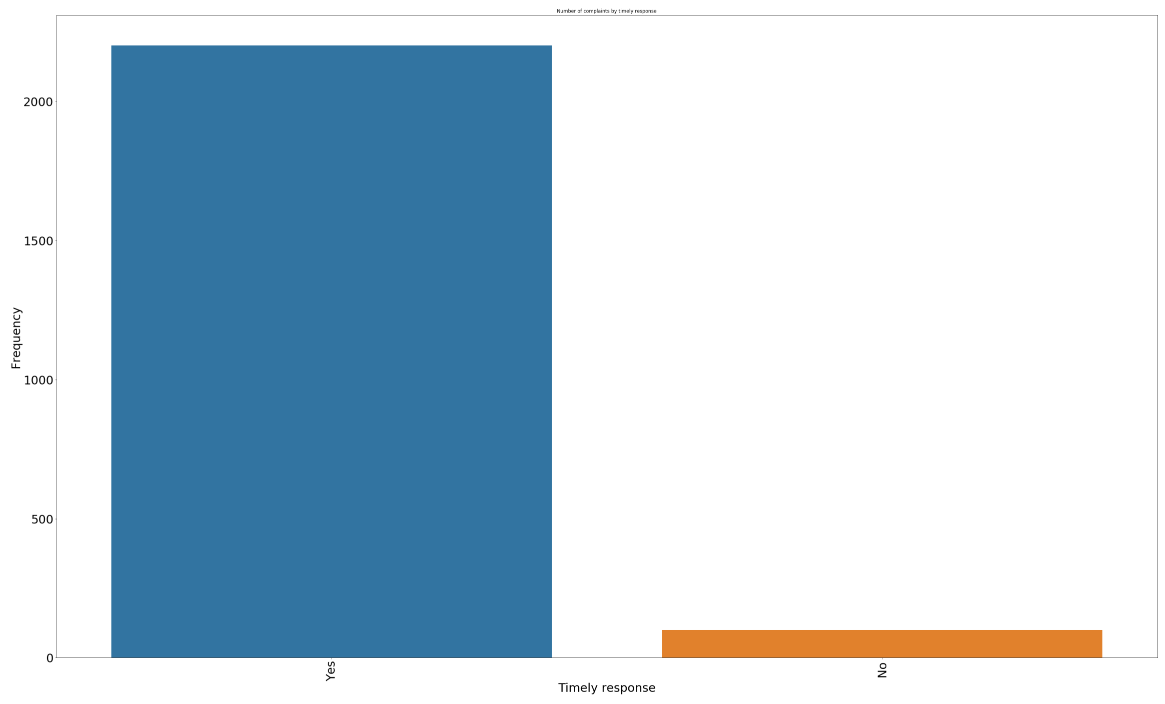
```
fig5 = sns.countplot(x='Sub-issue', data = complaints)
fig5.figure.set_size_inches(50,30)
fig5.set_xlabel("Sub-issue", fontsize=30)
fig5.set_ylabel("Frequency", fontsize=30)
plt.title("Number of complaints by sub-issue")
plt.xticks(rotation=90)
fig5.tick_params(labelsize = 30)
plt.show()
```



The "debt is not mine" and "debt was paid" follow on the above visualisation. There is a low amount of complaints relating to customer service which Santander should be encouraged by.

In [18]:

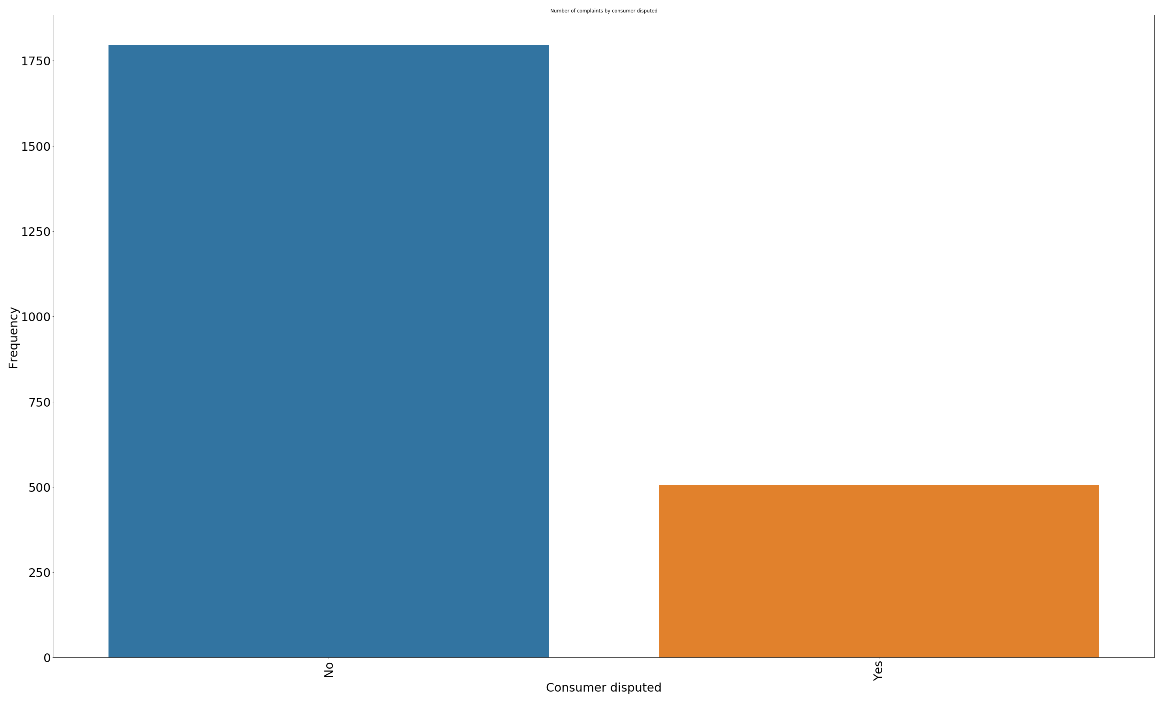
```
fig6 = sns.countplot(x='Timely response', data = complaints)
fig6.figure.set_size_inches(50,30)
fig6.set_xlabel("Timely response", fontsize=30)
fig6.set_ylabel("Frequency", fontsize=30)
plt.title("Number of complaints by timely response")
plt.xticks(rotation=90)
fig6.tick_params(labelsize = 30)
plt.show()
```



The large majority of complaints being responded to in a timely manner shows the process is running. I believe there are still too many "No". These deadlines should be met.

In [19]:

```
fig7 = sns.countplot(x='Consumer disputed', data = complaints)
fig7.figure.set_size_inches(50,30)
fig7.set_xlabel("Consumer disputed", fontsize=30)
fig7.set_ylabel("Frequency", fontsize=30)
plt.title("Number of complaints by consumer disputed")
plt.xticks(rotation=90)
fig7.tick_params(labelsize = 30)
plt.show()
```



The above shows me the two columns which are arguably the most useful, timely response and customer disputed. From a first glance it looks about one quarter of complaint outcomes are disputed. I think this is a higher than expected result. I will explore these columns relationships to other inputs to see if there is any obvious correlation.

The most common complaints are regarding 'attempts to collect on debt not owed' by a significant margin. This is clearly an area of process improvement that should be raised to the managers of this process.

In [20]:

```
# show the values in the timely response column
complaints['Timely response'].value_counts()
```

Out[20]:

```
Yes    2201
No      100
Name: Timely response, dtype: int64
```

In [21]:

```
# Percentage of cases that were not handled in a timely manner
P = 100/(100+2201)
P
```

Out[21]:

```
0.04345936549326380
```

- Only 4% of complaints were not handled in a timely manner.
- This is a good sign regarding customer service.

In [22]:

```
# show the values in the consumer disputed column
complaints['Consumer disputed'].value_counts()
```

Out[22]:

```
No     1795
Yes     506
Name: Consumer disputed, dtype: int64
```

In [23]:

```
# Percentage of cases in which the response was disputed by the customer
P = 506/(1705+506)
P
```

Out[23]:

```
0.22885572139303484
```

- 22% of all complaints are disputed.
  - My initial estimate based on the visualiation was slightly off.
  - The data contains no insight as to why these are disputed so any further commentary would be based on inference.

Analyise the data and tabulated analysis

Table formatting can help highlight the more obvious patterns in the dataset.

In [24]:

```
# new varable with issue and timely response.
CT = pd.crosstab(complaints['Issue'], complaints['Timely response'])
CT.sort_values('No', ascending = False)
```

Out[24]:

Timely response	No	Yes
Issue		
Cont'd attempts collect debt not owed	59	1041
Communication tactics	10	262
Disclosure verification of debt	9	328
False statements or representation	7	208
Taking/threatening an illegal action	7	149
Dealing with my lender or servicer	4	63
Can't repay my loan	2	24
Improper contact or sharing of info	2	123
Getting a loan	0	3

- I can see that there are not many complaints that did not get a timely response.
- I will concentrate on the customer disputed complaints.
- The most popular type of complaints are about attempts at collecting debt no owned as stated above.
    - This is a serious issue and can really affect a customer's trust in their bank.

In [25]:

```
# new varable with issue and consumer disputed.
CT = pd.crosstab(complaints['Issue'], complaints['Consumer disputed'])
CT.sort_values('Yes', ascending = False)
```

Out[25]:

Consumer disputed	**No**	**Yes**
Issue		
Cont'd attempts collect debt not owed	867	233
Disclosure verification of debt	250	87
Communication tactics	227	45
False statements or representation	170	45
Taking/threatening an illegal action	111	45
Improper contact or sharing of info	94	31
Dealing with my lender or servicer	54	13
Can't repay my loan	19	7
Getting a loan	3	0

In [26]:

```
P = 233/(867+233)
P
```

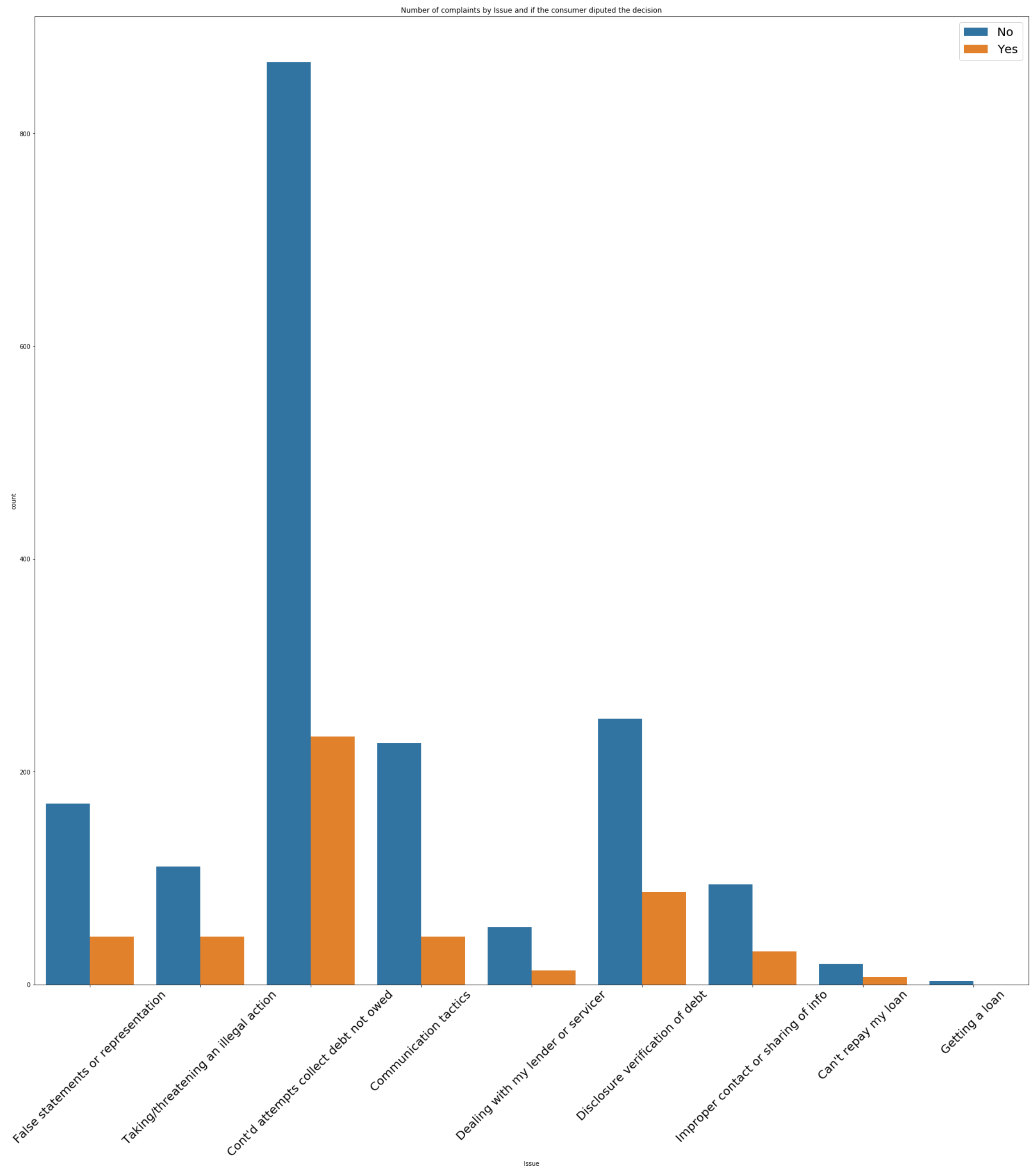
Out[26]:

```
0.21181818181818182
```

- 21% of the outcomes regarding the most complained about are disputed.
- This is in line with the overall complaints being disputed so I can take from this that complaint rates do not deviate from the norm.

In [27]:

```
fig8 = sns.countplot( x='Issue', hue= 'Consumer disputed', data = complaints)
fig8.figure.set_size_inches(30,30)
plt.xticks(fontsize = 20, rotation = 45)
plt.legend(loc = 'upper right', prop = {'size': 20}, fontsize = 20)
plt.title("Number of complaints by Issue and if the consumer diputed the decision")
plt.show()
```



I think the above plot is slightly unfair due to the vast amount on issues relating to illegal actions. Due to the axis scale it is distorting the results for other issues.

In [28]:

```
CT = pd.crosstab(complaints['Product'], complaints['Consumer disputed'])
CT.sort_values('Yes', ascending = False)
```

Out[28]:

Consumer disputed	No	Yes
Product		
Debt collection	1719	486
Student loan	76	20

In [29]:

```
SL = 20/(76+20)
SL
```

Out[29]:

```
0.20833333333333334
```

In [30]:

```
DC = 486/(1719+486)
DC
```

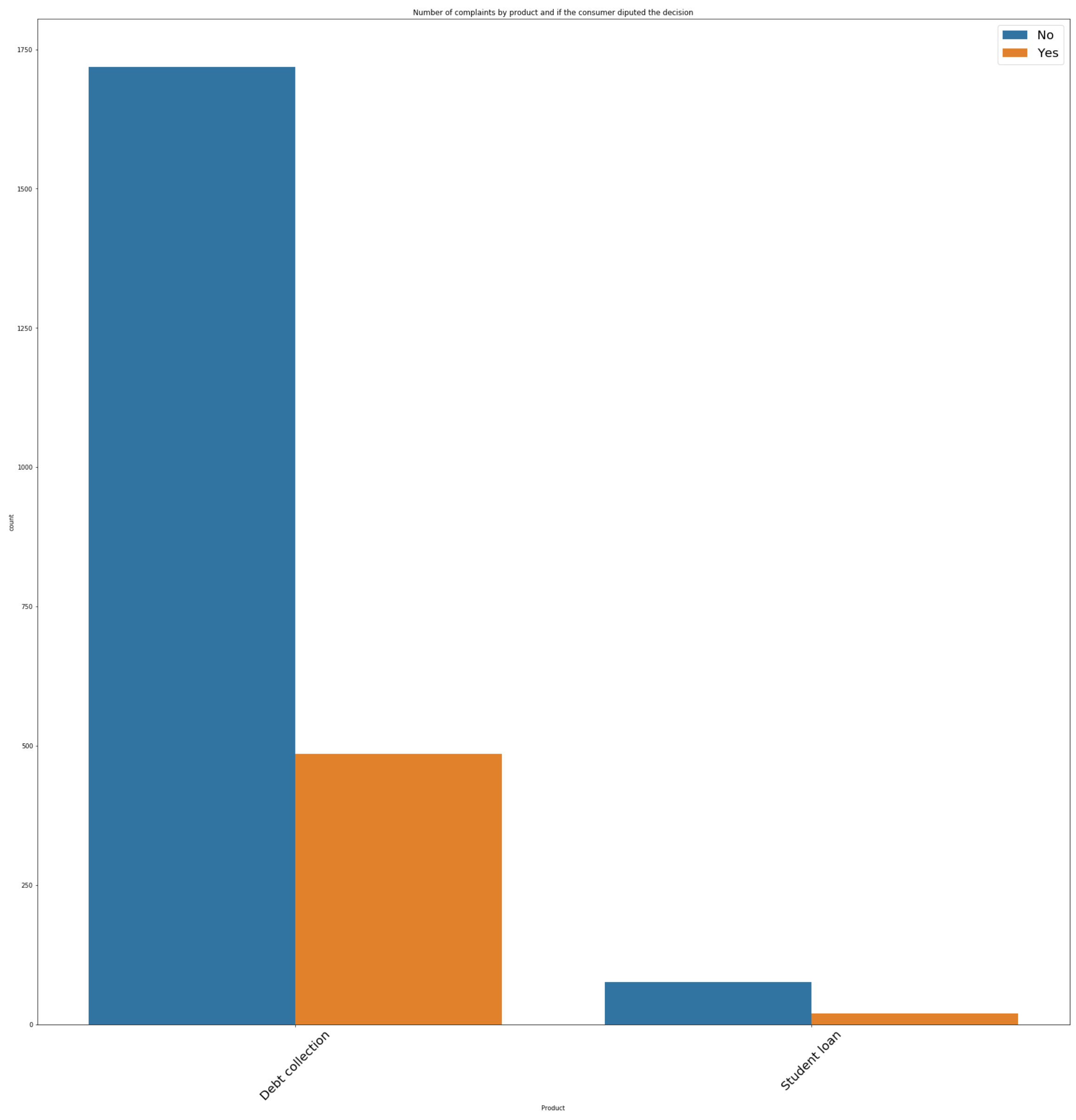
Out[30]:

```
0.22040816326530613
```

- There appears to be a slighly lower dispute rate for student loans than debt collection.

In [31]:

```
fig9 = sns.countplot( x='Product', hue= 'Consumer disputed', data = complaints)
fig9.figure.set_size_inches(30,30)
plt.xticks(fontsize = 20, rotation = 45)
plt.legend(loc = 'upper right', prop = {'size': 20})
plt.title("Number of complaints by product and if the consumer diputed the decision")
plt.show()
```



Student loans are one product offered whilst a number of products will fall under the debt collection complaint banner hence the distortion of the axis.

In [32]:

```
# show which sub-products had consumers dispute the decision
CT = pd.crosstab(complaints['Sub-product'], complaints['Consumer disputed'])
CT.sort_values('Yes', ascending = False)
```

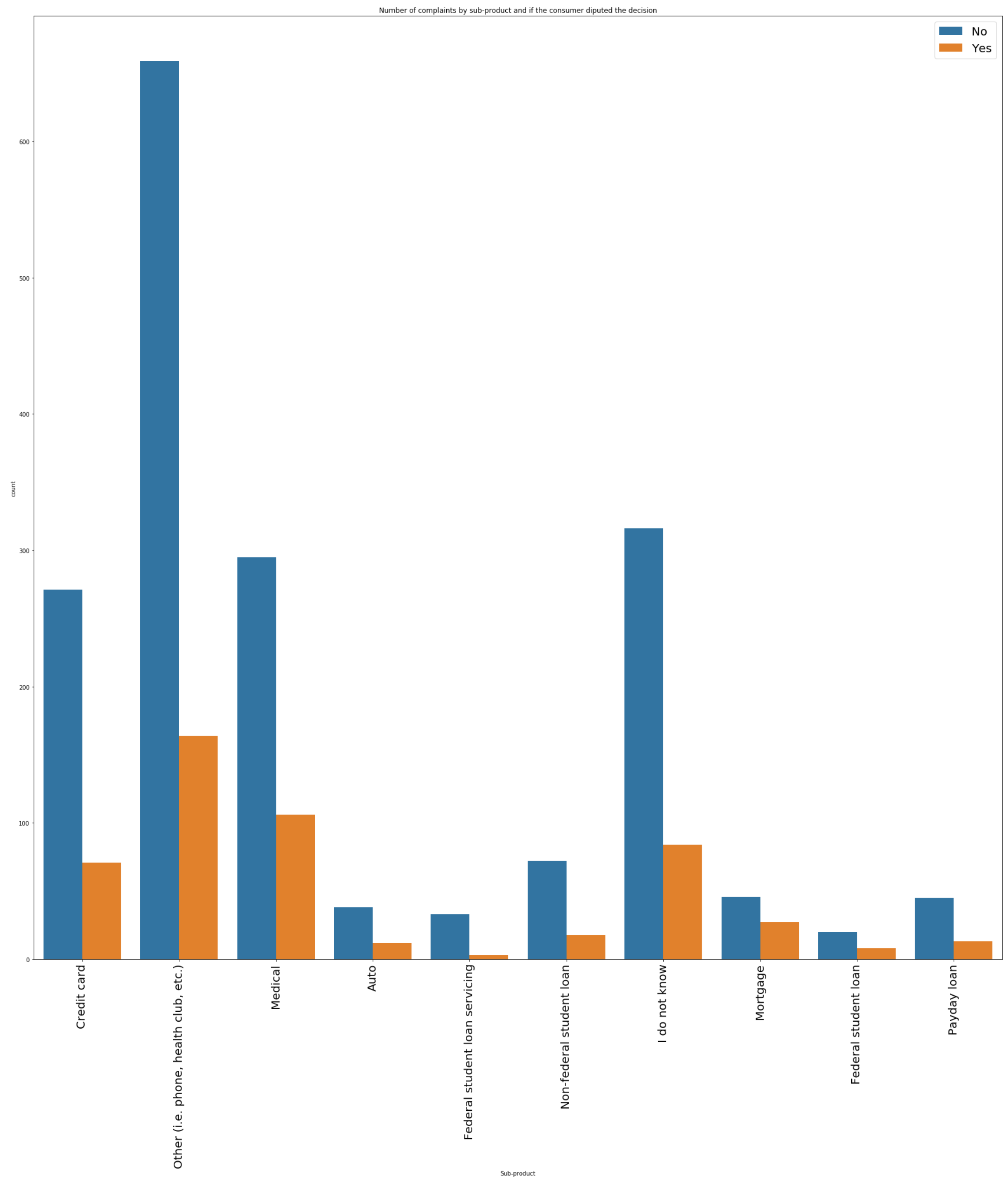
Out[32]:

Consumer disputed	No	Yes
Sub-product		
Other (i.e. phone, health club, etc.)	659	164
Medical	295	106
I do not know	316	84
Credit card	271	71
Mortgage	46	27
Non-federal student loan	72	18
Payday loan	45	13
Auto	38	12
Federal student loan	20	8
Federal student loan servicing	33	3

- There is a sub-product "I do not know".
- This data point that might have an effect on the results later as I will not be able to categorise these effectively.
    - It might impact the decision tree and is issue cause by the dataset itself.

In [33]:

```
fig10 = sns.countplot( x='Sub-product', hue= 'Consumer disputed', data = complaints)
fig10.figure.set_size_inches(30,30)
plt.xticks(fontsize = 20, rotation = 90)
plt.legend(loc = 'upper right', prop = {'size': 20})
plt.title("Number of complaints by sub-product and if the consumer diputed the decisio
n")
plt.show()
```



There is an interesting split in this above chart between federal and non-federal student loans. Having "I do not know" as a sub-product is an issue with the data. This should have been classified at source.

Correlation analysis

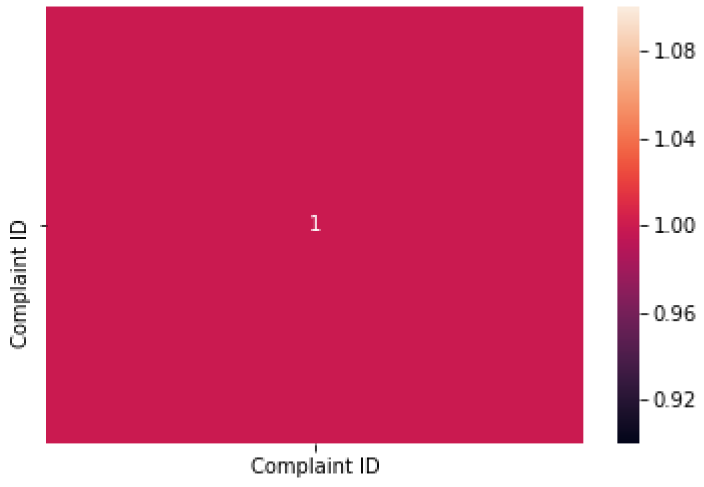
- I am going to look at the correlation between the variables.

In [34]:

```
sns.heatmap(complaints.corr(),  annot = True)
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2aa52828>
```



The ".corr" functionality only accepts numerical values so I need to factorise my data frame into a usable format.

In [35]:

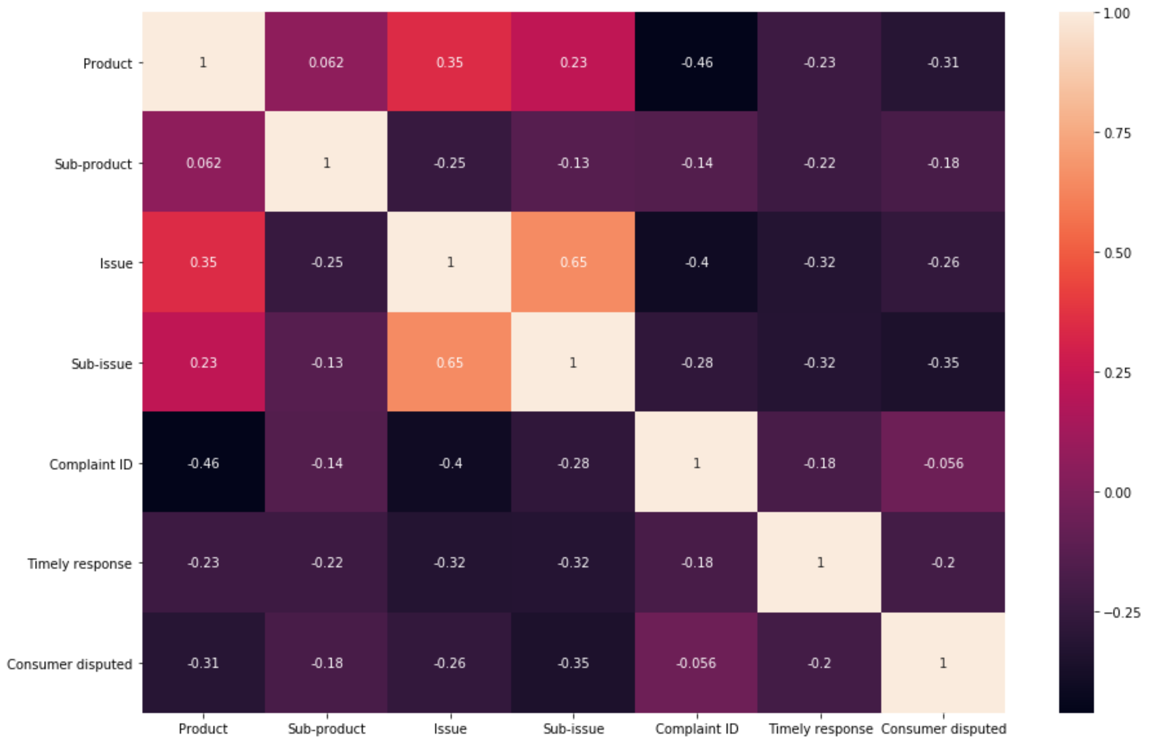
```
complaintsfactorize = complaints.apply(lambda x: x.factorize()[0]).corr()
```

In [36]:

```python
# heatmap of factorised inputs
plt.figure(figsize = (15,10))
sns.heatmap(complaintsfactorize.corr(),  annot = True)
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc49630>
```



There are a number of interesting correlations in this heat map but they do not seem that strong.

- The strongest correlation is between Issue and Sub-issue which makes sense considering the categorisation of the data.
  - Interestingly there is a slight negative correlation between the customer disputing and a timely response being received.
    - I would have expected this to be the opposite however as the two are unrelated it's easy to see why. Just because an outcome decision is late does not mean it is incorrect.

I can ignore the complaint ID fields as these unique values given to evert complaint.

r-coefficients and p-values

I am going to use researchpy to generate statistical information on my dataset.

In [37]:

```
# import researchpy for statistical analysis
import researchpy as rp
corr_type, corr_matrix, corr_ps = rp.corr_case(complaintsfactorize)
```

In [38]:

```
# show analysis
corr_ps
```

Out[38]:

	Product	Sub-product	Issue	Sub-issue	Complaint ID	Timely response	Consumer disputed
Product	0.0000	0.8945	0.4439	0.6236	0.2974	0.6258	0.5022
Sub-product	0.8945	0.0000	0.5952	0.7800	0.7583	0.6294	0.6939
Issue	0.4439	0.5952	0.0000	0.1145	0.3744	0.4784	0.5668
Sub-issue	0.6236	0.7800	0.1145	0.0000	0.5467	0.4893	0.4394
Complaint ID	0.2974	0.7583	0.3744	0.5467	0.0000	0.6941	0.9059
Timely response	0.6258	0.6294	0.4784	0.4893	0.6941	0.0000	0.6616
Consumer disputed	0.5022	0.6939	0.5668	0.4394	0.9059	0.6616	0.0000

In [39]:

```
#create corr table
corr_table = rp.corr_pair(complaintsfactorize)
```

In [40]:

```
# show the table
corr_table.head()
```

Out[40]:

	r value	p-value	N
Product & Sub-product	0.0623	0.8945	7
Product & Issue	0.3483	0.4439	7
Product & Sub-issue	0.2276	0.6236	7
Product & Complaint ID	-0.4614	0.2974	7
Product & Timely response	-0.2261	0.6258	7

As expected there is a strong p value between Product and Sub-issue.

The biggest one of note is the Product and Time response.
This potentially could be explored more as there would be some products which are taking a longer time to respond to. Going even further into the Sub-issue could provide more detailed insights.

In [41]:

```
# check the input types
corr_table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 21 entries, Product & Sub-product to Timely response & Consumer dis
puted
Data columns (total 3 columns):
r value    21 non-null object
p-value    21 non-null object
N          21 non-null int64
dtypes: int64(1), object(2)
memory usage: 1.3+ KB
```

The r-value and p-value are not numeric values so I am going to covert these so they are more usable.

In [42]:

```
# using pandas to convert this
corr_table['p-value'] = pd.to_numeric(corr_table['p-value'])
corr_table['r value'] = pd.to_numeric(corr_table['r value'])
```

In [43]:

```
#check the conversation has been successful
corr_table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 21 entries, Product & Sub-product to Timely response & Consumer dis
puted
Data columns (total 3 columns):
r value    21 non-null float64
p-value    21 non-null float64
N          21 non-null int64
dtypes: float64(2), int64(1)
memory usage: 1.3+ KB
```

In [44]:

```
# replotting the table for confirmation
corr_table.head()
```

Out[44]:

	r value	p-value	N
Product & Sub-product	0.0623	0.8945	7
Product & Issue	0.3483	0.4439	7
Product & Sub-issue	0.2276	0.6236	7
Product & Complaint ID	-0.4614	0.2974	7
Product & Timely response	-0.2261	0.6258	7

In [45]:

```
# highting the strongest r-values for potential exploration
corr_table[(corr_table['r value'] > 0.33)]
```

Out[45]:

	r value	p-value	N
Product & Issue	0.3483	0.4439	7
Issue & Sub-issue	0.6494	0.1145	7

In [46]:

```
# highlighting thhe strongest negitive r value
corr_table[(corr_table['r value']  < -0.33)]
```

Out[46]:

	r value	p-value	N
Product & Complaint ID	-0.4614	0.2974	7
Issue & Complaint ID	-0.3996	0.3744	7
Sub-issue & Consumer disputed	-0.3515	0.4394	7

In [47]:

```
# getting the dummie values to create a sparse matrix
cor_data = pd.get_dummies(complaints)
cor_data.head()
```

Out[47]:

	Complaint ID	Product_Debt collection	Product_Student loan	Sub-product_Auto	Sub-product_Credit card	product_F studen
514012	2446820	1	0	0	1	
514353	2447164	1	0	0	1	
515232	2445095	1	0	0	0	
515509	2442145	1	0	0	0	
515702	2441597	1	0	0	0	

5 rows × 62 columns

The sparse matrix that has been created has turned the values into Boolean (true or false) values which will allow me to run the classification model.

The Classification model

Split the data

I am splitting the data 75/25 to train the model and test against the dataset as this is a standard beginning split when creating a first model.

In [48]:

```
# dorp the unrequired columns from the cor_data
feature_cols = cor_data.drop(['Complaint ID','Consumer disputed_No','Consumer disputed_
Yes'], axis = 1)
```

In [49]:

```
# split the data into train and test populations
X = feature_cols
y = complaints['Consumer disputed']
X_train, X_test, y_train, y_test=train_test_split(X, y,
                                                  test_size = 0.25,
                                                  random_state = 123)
```

I am using the random states 123 for repeatable analysis and consistent results.

Setting up the model

The first model I am going to use is the random forest classifier with the criteria of entropy to classify the different elements.

In [50]:

```
clf = RandomForestClassifier(n_estimators = 500,criterion = 'entropy', max_depth = 5,ra
ndom_state = 123)
clf.fit(X_train,y_train)
```

Out[50]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entro
py',
            max_depth=5, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
            oob_score=False, random_state=123, verbose=0, warm_start=Fals
e)
```

In [51]:

```
#Classify the test subset using .predict()
y_pred = clf.predict(X_test)
```

In [52]:

```
#Calculate the accuracy using metrics.accuracy_score(y_test,y_pred)
print("Accuracy:", metrics.accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.7986111111111112
```

A 79% accuracy rating is very promising for the model.

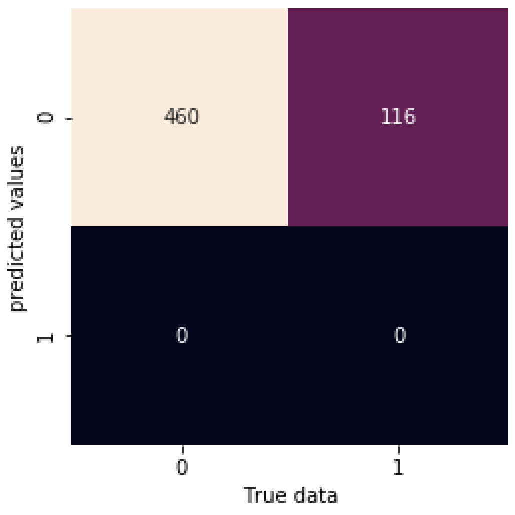
Changing the scope and input might advisedly effect the accuracy of the model.

In [53]:

```
# y_test is a dataframe and I am using theconfusion_matrix  function as y_test needs to
be converted to a list
from sklearn.metrics import confusion_matrix
y_true = y_test.tolist()
mat = confusion_matrix(y_true,y_pred)
sns.heatmap(mat.T, square =True, annot = True, fmt = 'd', cbar= False)
plt.xlabel('True data')
plt.ylabel('predicted values')
```

Out[53]:

```
Text(91.68, 0.5, 'predicted values')
```

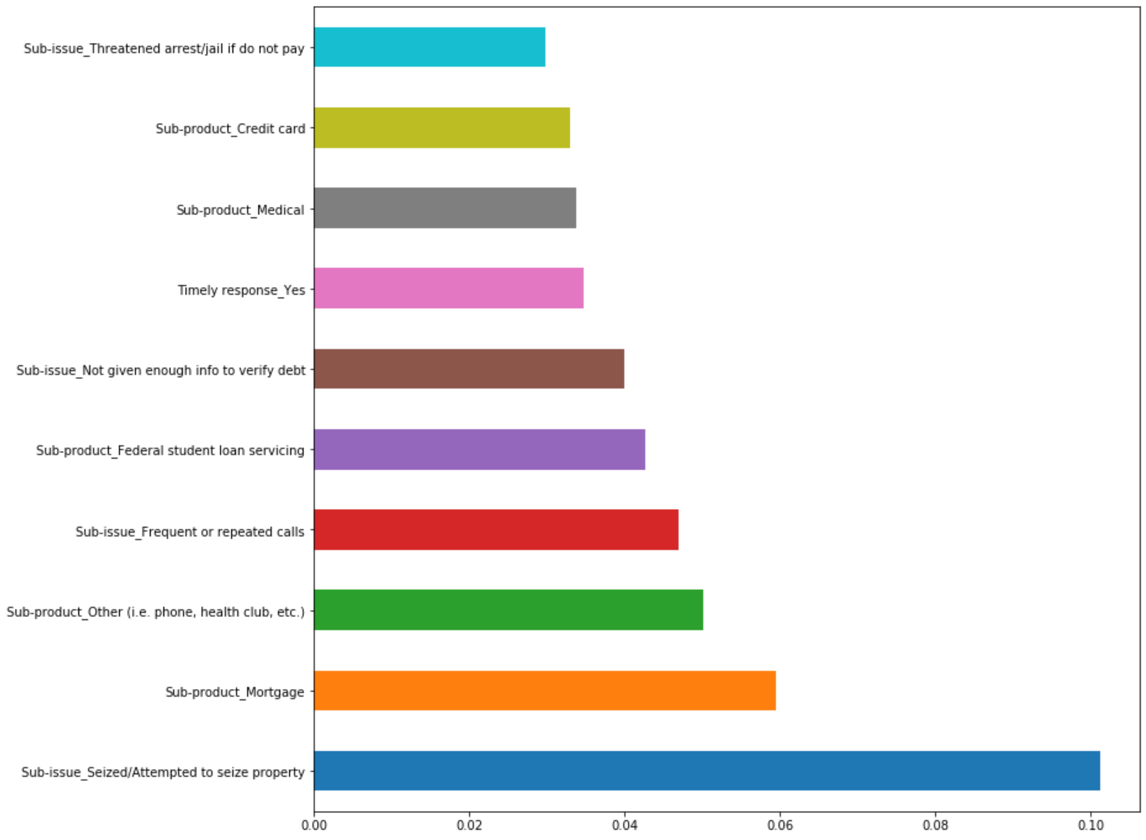


In [54]:

```
# visualising the most important features
feature_importances = pd.Series(clf.feature_importances_, index = X.columns)
feature_importances = feature_importances.sort_values(ascending = False).head(10)
feature_importances.plot(kind = 'barh', figsize = (12,12))
```

Out[54]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2cd5edd8>
```



Attempting to seize property is the most important feature. This result is not surprising as the repossession of property due to debt is a highly emotive topic and if done incorrectly can have a devastating impact on anyone's life. This is very related to the the issue of trying to collect on debt not owned.

Creating the deicsion tree

I am going to create a decision tree to visualise the model's categorisation.

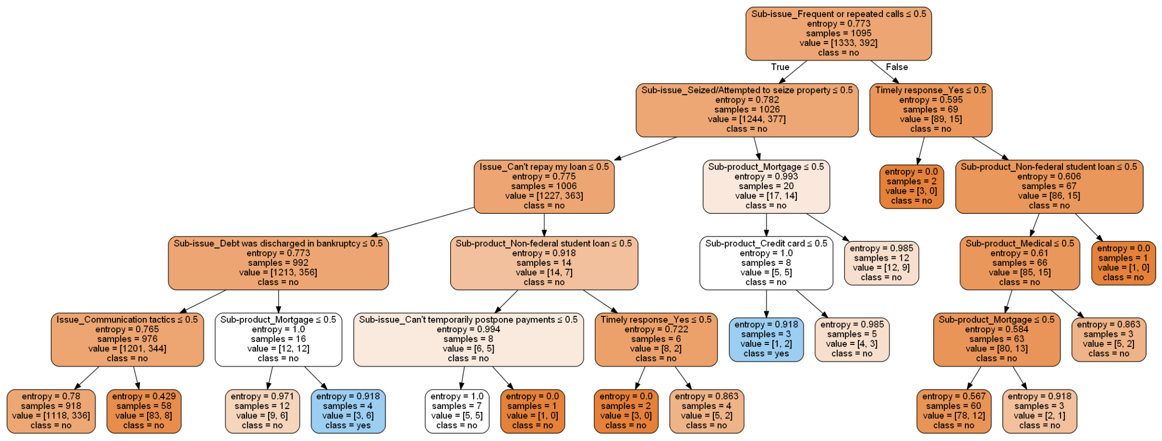
In [55]:

```
# importing the libaraies requred to create the decision tree
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

In [56]:

```
dot_data = StringIO()
estimator = clf.estimators_[5]
export_graphviz(estimator, out_file = dot_data,
                filled = True, rounded = True,
                special_characters = True, feature_names = X_train.columns,class_names
= ['no', 'yes'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[56]:


Sub-issue_Frequent or repeated calls ≤ 0.5 entropy = 0.773 samples = 1095 value = [1333, 392] class = no
True
False
Sub-issue_Seized/Attempted to seize property ≤ 0.5 entropy = 0.782 samples = 1026 value = [1244, 377] class = no
Timely response_Yes ≤ 0.5 entropy = 0.595 samples = 69 value = [89, 15] class = no
Issue_Can't repay my loan ≤ 0.5 entropy = 0.775 samples = 1006 value = [1227, 363] class = no
Sub-product_Mortgage ≤ 0.5 entropy = 0.993 samples = 20 value = [17, 14] class = no
entropy = 0.0 samples = 2 value = [3, 0] class = no
Sub-product_Non-federal student loan ≤ 0.5 entropy = 0.606 samples = 67 value = [86, 15] class = no
Sub-issue_Debt was discharged in bankruptcy ≤ 0.5 entropy = 0.773 samples = 992 value = [1213, 356] class = no
Sub-product_Non-federal student loan ≤ 0.5 entropy = 0.918 samples = 14 value = [14, 7] class = no
Sub-product_Credit card ≤ 0.5 entropy = 1.0 samples = 8 value = [5, 5] class = no
entropy = 0.985 samples = 12 value = [12, 9] class = no
Sub-product_Medical ≤ 0.5 entropy = 0.61 samples = 66 value = [85, 15] class = no
entropy = 0.0 samples = 1 value = [1, 0] class = no
Issue_Communication tactics ≤ 0.5 entropy = 0.765 samples = 976 value = [1201, 344] class = no
Sub-product_Mortgage ≤ 0.5 entropy = 1.0 samples = 16 value = [12, 12] class = no
Sub-issue_Can't temporarily postpone payments ≤ 0.5 entropy = 0.994 samples = 8 value = [6, 5] class = no
Timely response_Yes ≤ 0.5 entropy = 0.722 samples = 6 value = [8, 2] class = no
entropy = 0.918 samples = 3 value = [1, 2] class = yes
entropy = 0.985 samples = 5 value = [4, 3] class = no
Sub-product_Mortgage ≤ 0.5 entropy = 0.584 samples = 63 value = [80, 13] class = no
entropy = 0.863 samples = 3 value = [5, 2] class = no
entropy = 0.78 samples = 918 value = [1118, 336] class = no
entropy = 0.429 samples = 58 value = [83, 8] class = no
entropy = 0.971 samples = 12 value = [9, 6] class = no
entropy = 0.918 samples = 4 value = [3, 6] class = yes
entropy = 1.0 samples = 7 value = [5, 5] class = no
entropy = 0.0 samples = 1 value = [1, 0] class = no
entropy = 0.0 samples = 2 value = [3, 0] class = no
entropy = 0.863 samples = 4 value = [5, 2] class = no
entropy = 0.567 samples = 60 value = [78, 12] class = no
entropy = 0.918 samples = 3 value = [2, 1] class = no

Results and evaluation of the model

As a first model I believe this is a good first attempt. It has an accuracy rating above 75% and a decision tree which is readable and logically makes sense. I think the data set used was favourable for this after the nulls were dropped.

I think this model could be improved by focusing on the Sub-products and potentially isolating the companies.

By classifying the Sub-product and company it could be helpful when deciding where the company needs to focus its efforts on reducing the amount of challenges to resolutions, and help identify where work needs to be done toregards to replying to complaints in a more timely manner. This is an easily addressed problem and it can be fixed with just a little more additional effort.

The next steps would be using the full data set with the null values to see what effect that would have on the overall accuracy.

Evaluation

These results are very promising. The model accuracy being over 75% accuracy would be beneficial for the business in the complaints area. Next steps would involve expanding the dataset to make sure it stays accurate with more data. My area of the business could use a model like this for classifying the request data and overlying data for cases which are escalated. I can use this to see the areas of the business which are not satisfied with our service.

This could impact UBS' approach to client complaints. I could classify complaint and extract the areas where client satisfaction is lowest. Further investigate into this could identify the most common issues and be addressed by the business directly. I could continue this by gathering UBS exclusive data and adapting the model for this. This analysis has highlighted the need to make sure complaints are accurately recorded. Avoiding having classifications in the data as "other" will reduce the usefulness of the outcome due to being unable go into detail.