In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [6]:

```
df = pd.read_csv("emails.csv")
df.head()
```

Out[6]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	P
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	0	0	1	0	
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	0	0	0	0	
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	0	0	0	0	
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	

5 rows × 3002 columns

In [8]:

```
df=df.drop(columns='Email No.')
```

In [9]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
print(scaler.fit(df))
scaler.transform(df)
```

```
MinMaxScaler()
```

Out[9]:

```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.03809524, 0.09848485, 0.06705539, ..., 0.00877193, 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        1.        ],
       [0.00952381, 0.0530303 , 0.        , ..., 0.00877193, 0.        ,
        1.        ],
       [0.1047619 , 0.18181818, 0.01166181, ..., 0.        , 0.        ,
        0.        ]])
```

In [10]:

```
x=df.iloc[:,:-1]
y= df['Prediction']
```

In [11]:

```
x.head(10)
```

```
x.head(10)
```

Out[11]:

	the	to	ect	and	for	of	a	you	hou	in	...	enhancements	connevey	jay	valued	lay	infrastructure	military	allowi
0	0	0	1	0	0	0	2	0	0	0	...	0	0	0	0	0	0	0	
1	8	13	24	6	6	2	102	1	27	18	...	0	0	0	0	0	0	0	
2	0	0	1	0	0	0	8	0	0	4	...	0	0	0	0	0	0	0	
3	0	5	22	0	5	1	51	2	10	1	...	0	0	0	0	0	0	0	
4	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0	0	0	0	
5	4	5	1	4	2	3	45	1	0	16	...	0	0	0	0	0	0	0	
6	5	3	1	3	2	1	37	0	0	9	...	0	0	0	0	0	0	0	
7	0	2	2	3	1	2	21	6	0	2	...	0	0	0	0	0	0	0	
8	2	2	3	0	0	1	18	0	0	3	...	0	0	0	0	0	0	0	
9	4	4	35	0	1	0	49	1	16	9	...	0	0	0	0	0	0	0	

10 rows × 3000 columns

In [12]:

```
y.head()
```

Out[12]:

```
0    0
1    0
2    0
3    0
4    0
Name: Prediction, dtype: int64
```

In [13]:

```
df.shape
```

Out[13]:

```
(5172, 3001)
```

In [14]:

```
df.columns
```

Out[14]:

```
Index(['the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou', 'in',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3001)
```

In [16]:

```
df.groupby('Prediction').describe()
```

Out[16]:

	the								to		...	ff		dry		
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std
Prediction																
0	3672.0	6.673747	10.843067	0.0	1.0	3.0	8.0	210.0	3672.0	5.851307	...	1.0	35.0	3672.0	0.007353	0.1
1	1500.0	6.559333	13.708431	0.0	0.0	2.0	5.0	105.0	1500.0	7.012667	...	2.0	114.0	1500.0	0.006000	0.07

2 rows × 24000 columns

In [18]:

```
df0=df[df.Prediction==0]
df1=df[df.Prediction==1]
```

In [19]:

```
df0.head()
```

Out[19]:

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prec
0	0	0	1	0	0	0	2	0	0	0	...	0	0	0	0	0	0	0	0	0	
1	8	13	24	6	6	2	102	1	27	18	...	0	0	0	0	0	0	0	1	0	
2	0	0	1	0	0	0	8	0	0	4	...	0	0	0	0	0	0	0	0	0	
3	0	5	22	0	5	1	51	2	10	1	...	0	0	0	0	0	0	0	0	0	
4	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0	0	0	0	1	0	

5 rows × 3001 columns

In [20]:

```
df1.head()
```

Out[20]:

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Pre
5	4	5	1	4	2	3	45	1	0	16	...	0	0	0	0	0	0	0	0	0	
7	0	2	2	3	1	2	21	6	0	2	...	0	0	0	0	0	0	0	1	0	
16	3	1	2	2	0	1	17	0	0	1	...	0	0	0	0	0	0	0	1	0	
17	36	21	6	14	7	17	194	25	5	59	...	0	0	0	0	0	0	0	3	0	
25	12	53	2	14	18	14	287	0	2	86	...	0	0	0	0	0	0	0	6	0	

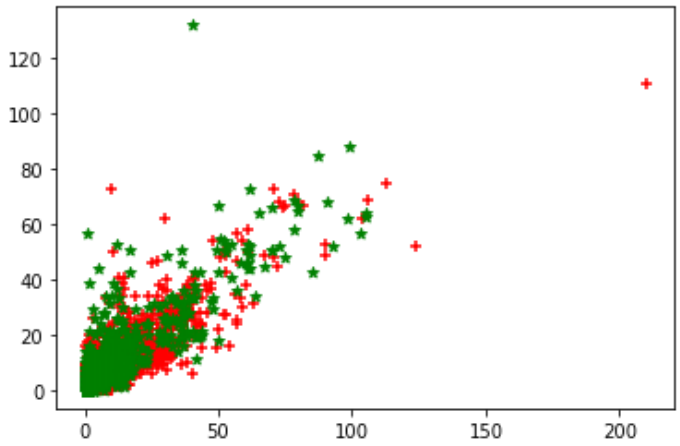
5 rows × 3001 columns

In [21]:

```
plt.scatter(df0['the'], df0['to'],color='red',marker='+')
plt.scatter(df1['the'], df1['to'],color='green',marker='*')
```

Out[21]:

```
<matplotlib.collections.PathCollection at 0x2238bceb790>
```



Feature Selection chi2

In [22]:

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

In [23]:

```
ordered_rank_feature = SelectKBest(score_func=chi2,k=7)
ordered_feature = ordered_rank_feature.fit(x,y)
ordered_feature
```

Out[23]:

```
SelectKBest(k=7, score_func=<function chi2 at 0x000002238C5A1DC0>)
```

In [24]:

```
dfscores=pd.DataFrame(ordered_feature.scores_,columns=['Score'])
dfcolumns=pd.DataFrame(x.columns)
```

In [25]:

```
dfscores
```

Out[25]:

	Score
0	2.099367
1	232.118330
2	2916.057301
3	803.818986
4	0.348715
...	...
2995	16.400743
2996	63.829543
2997	0.843236
2998	802.004211
2999	0.280059

3000 rows × 1 columns

In [26]:

```
dfcolumns
```

Out[26]:

	0
0	the
1	to
2	ect
3	and
4	for
...	...
2995	infrastructure

	0
2996	military
2997	allowing
2998	ff
2999	dry

3000 rows × 1 columns

In [27]:

```
features_rank = pd.concat([dfcolumns,dfscores],axis=1)
```

In [28]:

```
features_rank.columns=['Features','Score']
features_rank
```

Out[28]:

	Features	Score
0	the	2.099367
1	to	232.118330
2	ect	2916.057301
3	and	803.818986
4	for	0.348715
...	...	...
2995	infrastructure	16.400743
2996	military	63.829543
2997	allowing	0.843236
2998	ff	802.004211
2999	dry	0.280059

3000 rows × 2 columns

In [29]:

```
features_rank.nlargest(10,'Score')
```

Out[29]:

	Features	Score
14	i	20933.845216
23	s	9905.907062
173	r	9581.168541
6	a	8297.355495
138	o	7767.524289
275	n	6818.365184
40	e	6309.085868
129	p	6102.448757
54	t	5700.608037
45	d	4752.245232

MultinomialNB

In [30]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.20, random_state =
0)
```

In [31]:

```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
```

In [32]:

```
classifier.fit(x_train,y_train)
```

Out[32]:

```
MultinomialNB()
```

In [33]:

```
classifier.score(x_test,y_test)
```

Out[33]:

```
0.9449275362318841
```

Classification Report

In [34]:

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(x_train)
```

In [35]:

```
print(classification_report(y_train, pred))
print()
print('Confusion Matrix: \n', confusion_matrix(y_train, pred))
print()
print('Accuracy: ', accuracy_score(y_train, pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.95      0.96      2922
           1       0.89      0.94      0.92      1215

    accuracy                           0.95      4137
   macro avg       0.93      0.95      0.94      4137
weighted avg       0.95      0.95      0.95      4137


Confusion Matrix:
 [[2777  145]
 [  68 1147]]

Accuracy:  0.9485134155184917
```

Linear Regression

In [36]:

```
from sklearn.linear_model import LinearRegression
 model = LinearRegression()
 model.fit(x_train,y_train)
```

Out[36]:

```
LinearRegression()
```

In [37]:

```
y_pred =model.predict(x_test)
y_pred
```

Out[37]:

```
array([ 0.88013538, -0.45459323,  0.14351809, ...,  0.30671957,
       -2.59940637,  0.03559231])
```

In [38]:

```
x_pred=model.predict(x_train)
x_pred
```

Out[38]:

```
array([ 1.00328913,  1.00836858, -0.01771057, ...,  1.01227394,
       -0.01276006,  0.99494817])
```

In [39]:

```
model.score(x_test,y_test)
```

Out[39]:

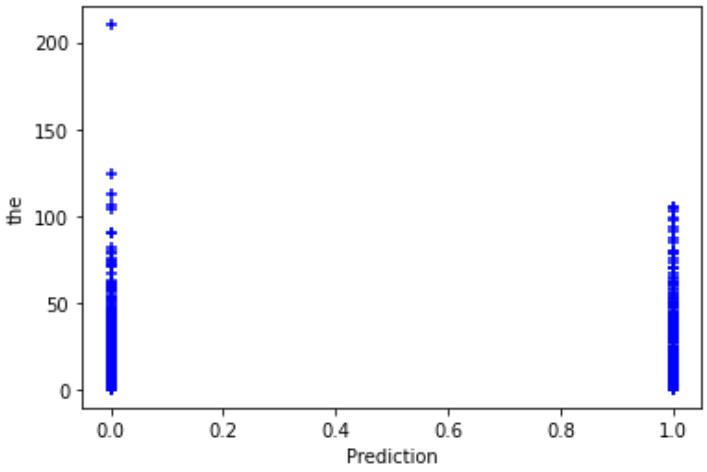
```
-50390.12746980952
```

In [40]:

```
plt.xlabel('Prediction')
plt.ylabel('the')
plt.scatter(df['Prediction'], df['the'],color='blue',marker='+')
```

Out[40]:

```
<matplotlib.collections.PathCollection at 0x2238ba2ad60>
```

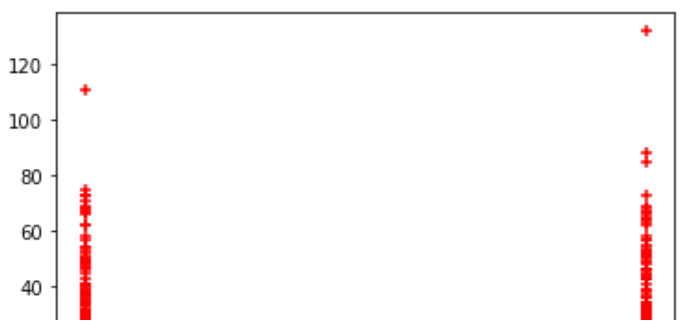


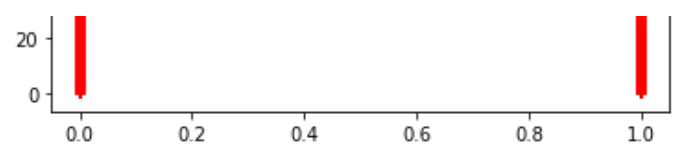
In [41]:

```
plt.scatter(df['Prediction'], df['to'],color='red',marker='+')
```

Out[41]:

```
<matplotlib.collections.PathCollection at 0x2238bc12bb0>
```



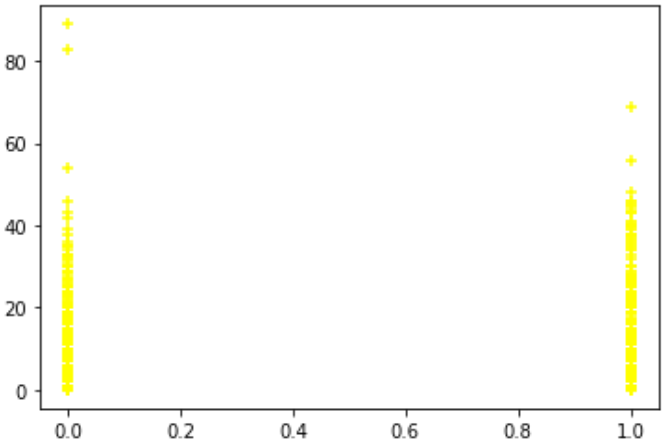


In [42]:

```
plt.scatter(df['Prediction'], df['and'],color='yellow',marker='+')
```

Out[42]:

<matplotlib.collections.PathCollection at 0x2238bb64b20>

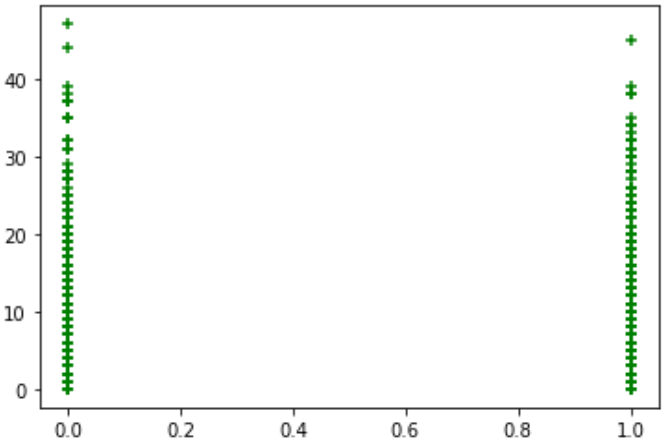


In [43]:

```
plt.scatter(df['Prediction'], df['for'],color='green',marker='+')
```

Out[43]:

<matplotlib.collections.PathCollection at 0x2238bb3fa90>

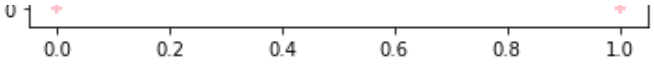


In [44]:

```
plt.scatter(df['Prediction'], df['you'],color='pink',marker='+')
```

Out[44]:

<matplotlib.collections.PathCollection at 0x2238b8eba00>

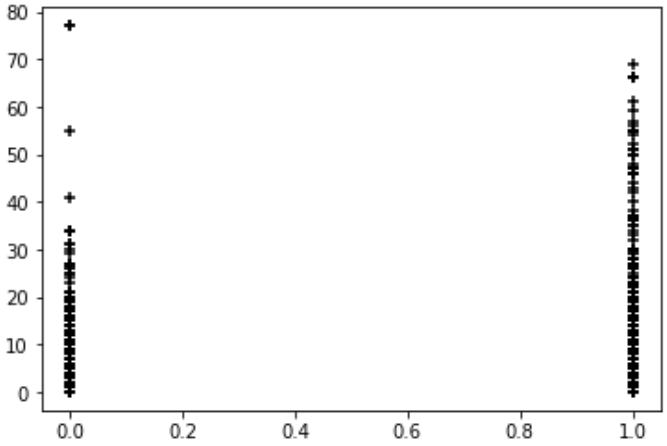


In [46]:

```
plt.scatter(df['Prediction'], df['of'],color='black',marker='+')
```

Out[46]:

```
<matplotlib.collections.PathCollection at 0x2238b5bb820>
```

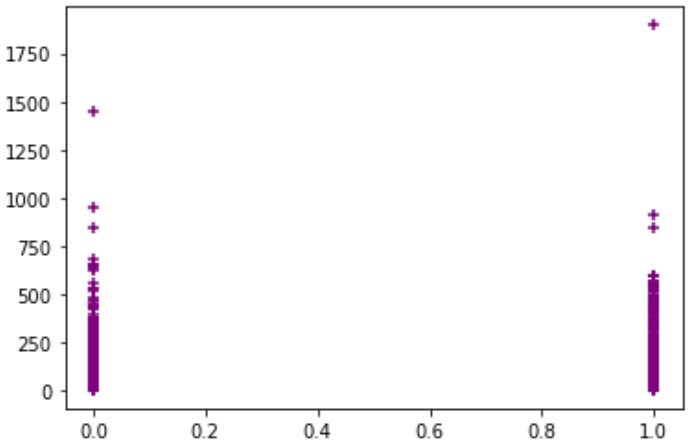


In [47]:

```
plt.scatter(df['Prediction'], df['a'],color='purple',marker='+')
```

Out[47]:

```
<matplotlib.collections.PathCollection at 0x2238b58bf10>
```

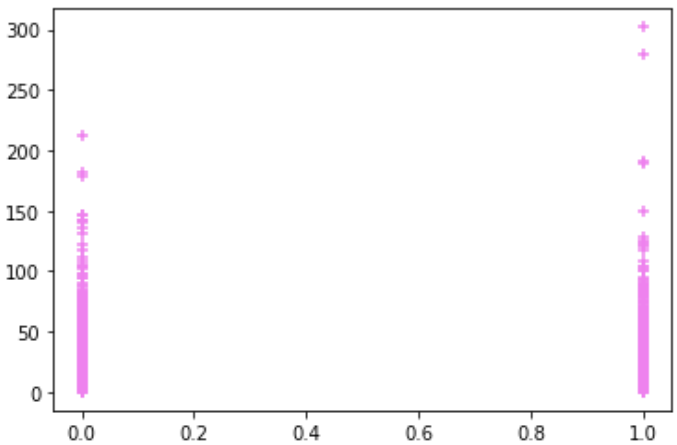


In [49]:

```
plt.scatter(df['Prediction'], df['on'],color='violet',marker='+')
```

Out[49]:

```
<matplotlib.collections.PathCollection at 0x2238bf5cfa0>
```

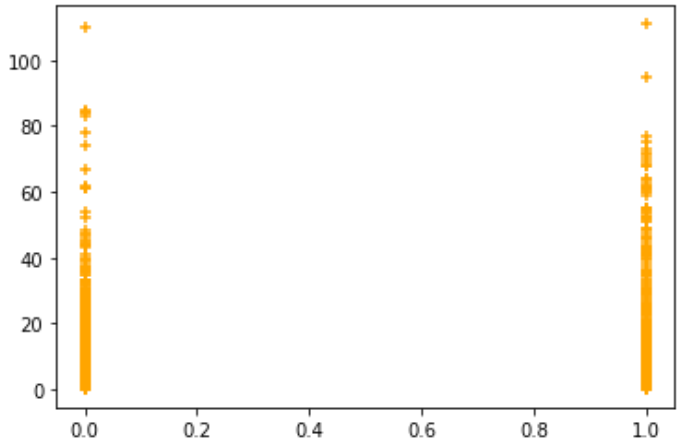


In [51]:

```
plt.scatter(df['Prediction'], df['is'],color='orange',marker='+')
```

Out[51]:

```
<matplotlib.collections.PathCollection at 0x2238c57c190>
```

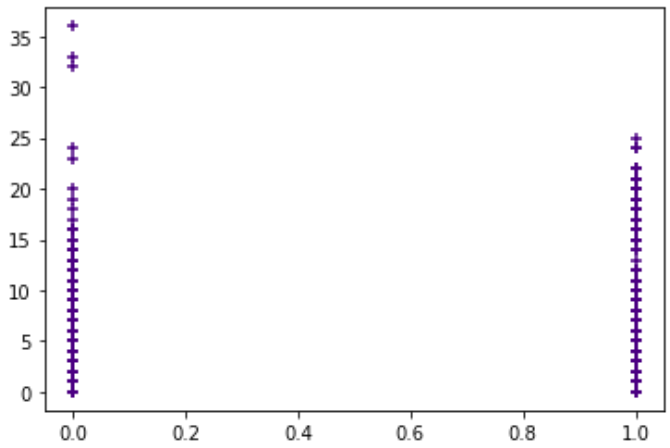


In [52]:

```
plt.scatter(df['Prediction'], df['this'],color='indigo',marker='+')
```

Out[52]:

```
<matplotlib.collections.PathCollection at 0x2238b808730>
```

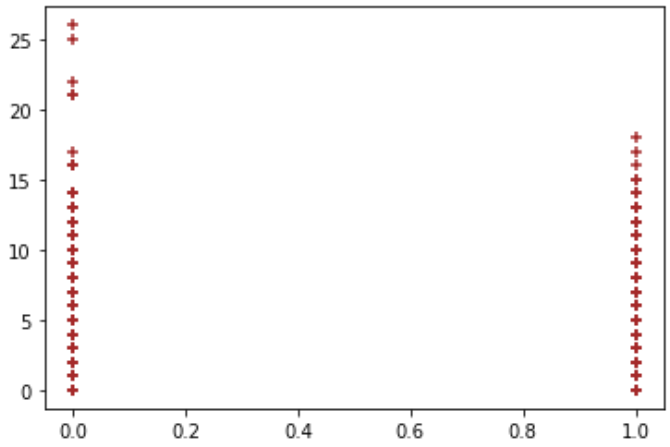


In [53]:

```
plt.scatter(df['Prediction'], df['that'],color='brown',marker='+')
```

Out[53]:

```
<matplotlib.collections.PathCollection at 0x2238bc43790>
```



Logistic Regression

In [54]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [55]:

```
model.fit(x_train, y_train)
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[55]:

```
LogisticRegression()
```

In [56]:

```
model.predict(x_test)
```

Out[56]:

```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

In [57]:

```
model.score(x_test, y_test)
```

Out[57]:

```
0.9632850241545894
```

Random forest

In [58]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)
```

Out[58]:

```
RandomForestClassifier()
```

In [59]:

```
model.score(x_test, y_test)
```

Out[59]:

```
0.9710144927536232
```

In [60]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=80, criterion='gini')
model.fit(x_train, y_train)
```

Out[60]:

```
RandomForestClassifier(n_estimators=80)
```

In [61]:

In [61]:

```
model.score(x_test, y_test)
```

Out[61]:

```
0.9642512077294686
```

In [62]:

```
y_pred = model.predict(x_test)
```

In [63]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
```

In [64]:

```
cm
```

Out[64]:

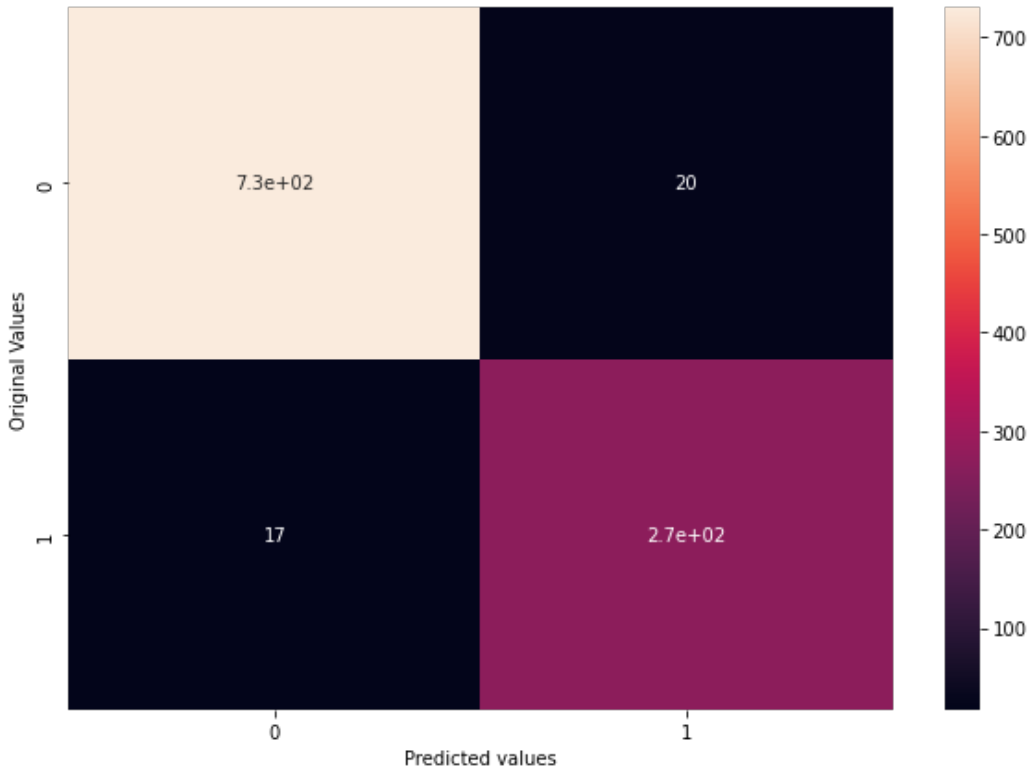
```
array([[730,  20],
       [ 17, 268]], dtype=int64)
```

In [65]:

```
import seaborn as sns
plt.figure(figsize=(10,7))
sns.heatmap(cm,annot=True)
plt.xlabel('Predicted values')
plt.ylabel('Original Values')
```

Out[65]:

```
Text(69.0, 0.5, 'Original Values')
```



Support Vector Machine

In [66]:

```
from sklearn.svm import SVC
```

In [67]:

```
model = SVC()
```

In [68]:

```
model.fit(x_train, y_train)
```

Out[68]:

```
SVC()
```

In [69]:

```
model.score(x_test,y_test)
```

Out[69]:

```
0.7951690821256039
```

In [70]:

```
Y_pred = model.predict(x_test)
```

In [71]:

```
from sklearn.metrics import confusion_matrix
c_m = confusion_matrix(y_test,Y_pred)
```

In [72]:

```
c_m
```

Out[72]:

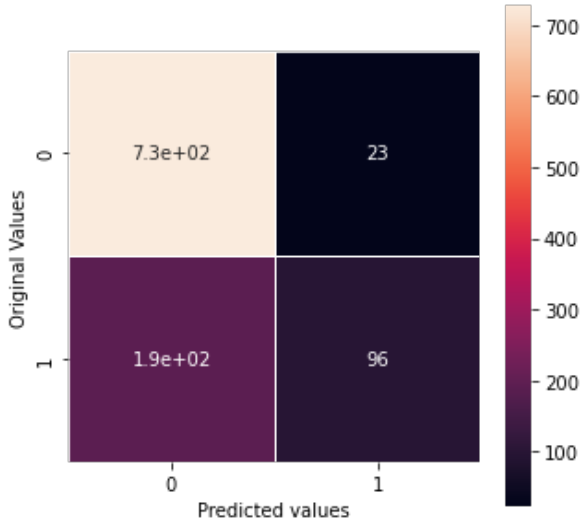
```
array([[727,  23],
       [189,  96]], dtype=int64)
```

In [73]:

```
plt.figure(figsize=(5,5))
sns.heatmap(c_m,annot=True, linewidth=.5, square = True)
plt.xlabel('Predicted values')
plt.ylabel('Original Values')
```

Out[73]:

```
Text(24.0, 0.5, 'Original Values')
```



Decision Tree

In [74]:

```
x = df.iloc[:,:-1]
```

In [75]:

```
y = df.iloc[:,:1]
```

In [76]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

In [77]:

```
from sklearn import tree
model = tree.DecisionTreeRegressor()
```

In [78]:

```
model.fit(x_train, y_train)
```

Out[78]:

```
DecisionTreeRegressor()
```

In [79]:

```
model.score(x_test,y_test)
```

Out[79]:

```
0.9150524627411312
```

In [80]:

```
model.predict(x_test[:6])
```

Out[80]:

```
array([1., 1., 0., 5., 0., 0.])
```

In [81]:

```
x_test[:8]
```

Out[81]:

	the	to	ect	and	for	of	a	you	hou	in	...	enhancements	connevey	jay	valued	lay	infrastructure	military	allow
3893	1	3	2	0	0	0	15	2	0	2	...	0	0	0	0	0	0	0	
3973	1	3	2	0	2	0	15	0	0	2	...	0	0	0	0	0	0	0	
4849	0	0	3	2	0	5	18	2	0	7	...	0	0	0	0	0	0	0	
883	5	0	1	0	3	1	17	1	1	0	...	0	0	0	0	0	0	0	
3955	0	1	1	0	0	0	2	0	0	1	...	0	0	0	0	0	0	0	
749	0	0	1	0	0	0	5	0	0	0	...	0	0	0	0	0	0	0	
4667	0	0	1	0	0	0	18	0	0	3	...	0	0	0	0	0	0	0	
818	2	1	1	1	0	1	17	0	0	4	...	0	0	0	0	0	0	0	

8 rows × 3000 columns

In [82]:

```
y_test[:8]
```

Out[82]:

	the
3893	1

3973	the
4849	0
883	5
3955	0
749	0
4667	0
818	2

Lasso Regression

In [83]:

```
from sklearn.linear_model import Lasso
lasso_model = Lasso()
```

In [84]:

```
lasso_model.fit(x_train, y_train)
```

Out[84]:

```
Lasso()
```

In [85]:

```
lasso_model.score(x_test,y_test)
```

Out[85]:

```
0.9998407107568604
```

In [86]:

```
lasso_model.score(x_train,y_train)
```

Out[86]:

```
0.9998055416381715
```

Ridge Regression

In [87]:

```
from sklearn.linear_model import Ridge
ridge_model = Ridge()
```

In [88]:

```
ridge_model.fit(x_train,y_train)
```

Out[88]:

```
Ridge()
```

In [89]:

```
ridge_model.score(x_test, y_test)
```

Out[89]:

```
0.9999996440393976
```

In [90]:

```
ridge_model.score(x_train,y_train)
```

Out[90]:

0.9999999917774038