Step 1 - add in relevant python libraries

In [67]:

```
import pandas as pd
import numpy as np

from astropy.table import Table
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import seaborn as sns
from sklearn.metrics import confusion_matrix
from matplotlib import pyplot as plt

#stars as 0, galaxies as 1
```

Step 2 - read in file information

In [48]:

```
filename = 'sgsep_cosmos_tests_v2.fits'
data = Table.read(filename, format='fits')
df = data.to_pandas()

#0 = galaxy, 1 = star
```

In [49]:

```
df.columns #these are all of the columns
```

Out[49]:

```
Index(['COADD_OBJECTS_ID', 'RA', 'DEC', 'MAG_AUTO_G', 'MAG_AUTO_R',
       'MAG_AUTO_I', 'MAG_AUTO_Z', 'MAG_AUTO_Y', 'MAGERR_AUTO_G',
       'MAGERR_AUTO_R', 'MAGERR_AUTO_I', 'MAGERR_AUTO_Z', 'MAGERR_AUTO_Y',
       'MAG_CM_MOF_G', 'MAG_CM_MOF_R', 'MAG_CM_MOF_I', 'MAG_CM_MOF_Z',
       'MAG_PSF_MOF_G', 'MAG_PSF_MOF_R', 'MAG_PSF_MOF_I', 'MAG_PSF_MOF_Z',
       'CONCENTRATION_MOF_G', 'CONCENTRATION_MOF_R', 'CONCENTRATION_MOF_I',
       'CONCENTRATION_MOF_Z', 'CLASS_STAR_I', 'SPREAD_MODEL_I',
       'SPREADERR_MODEL_I', 'CM_T', 'CM_T_ERR', 'MCAL_RATIO', 'HB_PROB',
       'TRUE_CLASS'],
      dtype='object')
```

In [50]:

```
df.head()
```

Out[50]:

	COADD_OBJECTS_ID	RA	DEC	MAG_AUTO_G	MAG_AUTO_R	MAG_AUTO_I	MAG_AUTO_Z	MAG_AUTO_Y	MAGERR_AU
0	3172103719	149.583228	1.801492	24.207701	23.973700	24.034401	23.503401	99.000000	0
1	3172103721	149.591019	1.801438	23.876200	23.272200	23.363100	22.822001	23.277201	0
2	3172103724	149.655285	1.801508	25.826000	24.482901	23.968201	24.639601	24.629900	0
3	3172103725	149.658410	1.801584	24.312500	23.829700	24.412901	23.942699	23.140499	0
4	3172103732	149.663295	1.801699	25.224300	23.979099	23.620399	24.607800	22.607901	0

5 rows × 33 columns

Step 3 - clean dataframe

In [51]:

```
df_properties = df.iloc[:,3:30] #photometric properties
#another way to do it is: df_properties = df.drop(['COADD_OBJECTS_ID','RA','DEC'],axis='columns')
y = df['TRUE_CLASS']
```

In [73]:

```
y.value_counts()
```

Out[73]:

```
0     103914
1      12113
Name: TRUE_CLASS, dtype: int64
```

In [52]:

```
df_properties.head()
```

Out[52]:

	MAG_AUTO_G	MAG_AUTO_R	MAG_AUTO_I	MAG_AUTO_Z	MAG_AUTO_Y	MAGERR_AUTO_G	MAGERR_AUTO_R	MAGERR_AUTO_I
0	24.207701	23.973700	24.034401	23.503401	99.000000	0.1224	0.1181	0.2037
1	23.876200	23.272200	23.363100	22.822001	23.277201	0.1307	0.0979	0.1362
2	25.826000	24.482901	23.968201	24.639601	24.629900	0.4103	0.1210	0.1142
3	24.312500	23.829700	24.412901	23.942699	23.140499	0.1412	0.0954	0.2471
4	25.224300	23.979099	23.620399	24.607800	22.607901	0.3086	0.1110	0.1202

5 rows × 27 columns

In [53]:

```
len(df_properties)
```

Out[53]:

```
116027
```

Step 4 - train and test the data

In [54]:

```
X_train, X_test, y_train, y_test = train_test_split(df_properties, y, test_size=0.2)

#X_train = the first 80% of the df_properties
#y_train is the first 80% of y which is the answers for the first 80% of df_properties

#X_test = 20% of df_properties that we want to test the model
#y_test = 20% of the true answers for the 20% of df_properties
```

In [55]:

```
#creating the model instance

clf = RandomForestClassifier(n_estimators=100,max_depth=2)
```

In [56]:

```
#training the model; give it all of the properties information (X), and the subsequent answers for that informa
#so that it learns everything

clf.fit(X_train,y_train)
```

Out[56]:

```
RandomForestClassifier(max_depth=2)
```

Step 5 - check accuracy of model

In [59]:

```
y_pred = clf.predict(X_test)
```

In [62]:

```
#accuracy score of the model; how accurate was the model in predicting whether it was a star or galaxy

str(round((accuracy_score(y_test, y_pred))*100))+'%'
```

Out[62]:

```
'96%'
```

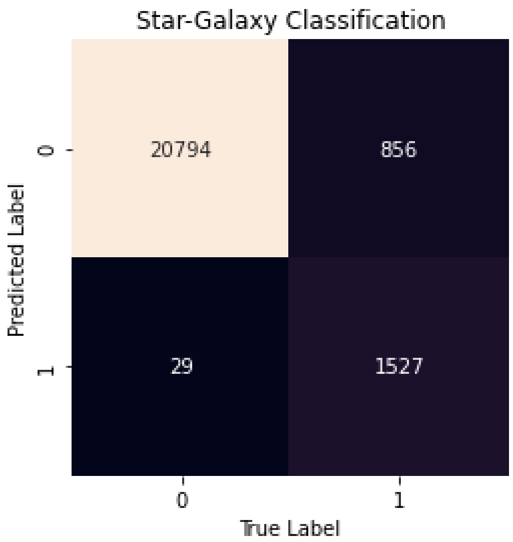
Step 6 - building confusion matrix

In [71]:

```
mat = confusion_matrix(y_test,y_pred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('True Label')
plt.ylabel('Predicted Label')
title = 'Star-Galaxy Classification'
plt.title(title)
plt.show()

TP, FP, FN, TN = float(mat[0][0]),float(mat[0][1]),float(mat[1][0]),float(mat[1][1])

print ("Completeness/Precision:", round((TP/(TP+FN)),3)) #sensitivity/true positive rate
print ("Purity:", round((TP/(TP+FP)),3)) #precision
gal_cont = round((FP/(FP+TP)),3)*100
print ("Galaxy Contamination:", gal_cont,'%') #when star is misclassified as galaxy
```



```
Completeness/Precision: 0.96
Purity: 0.999
Galaxy Contamination: 0.1 %
```