In [1]:
```python
import pandas as pd
```

In [2]:
```python
import matplotlib.pyplot as plt
```

In [3]:
```python
import seaborn as sb
```

In [4]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```python
df = pd.read_csv('C:/Users/gl/star_classification.csv')
df.head(6)
```

Out[5]:

	obj_ID	alpha	delta	u	g	r	i	z	run_ID	rerun_ID	cam_col	field_ID	spec_obj_ID	class	redshift	plate	MJD	fiber_ID
0	1.237661e+18	135.689107	32.494632	23.87882	22.27530	20.39501	19.16573	18.79371	3606	301	2	79	6.543777e+18	GALAXY	0.634794	5812	56354	171
1	1.237665e+18	144.826101	31.274185	24.77759	22.83188	22.58444	21.16812	21.61427	4518	301	5	119	1.176014e+19	GALAXY	0.779136	10445	58158	427
2	1.237661e+18	142.188790	35.582444	25.26307	22.66389	20.60976	19.34857	18.94827	3606	301	2	120	5.152200e+18	GALAXY	0.644195	4576	55592	299
3	1.237663e+18	338.741038	-0.402828	22.13682	23.77656	21.61162	20.50454	19.25010	4192	301	3	214	1.030107e+19	GALAXY	0.932346	9149	58039	775
4	1.237680e+18	345.282593	21.183866	19.43718	17.58028	16.49747	15.97711	15.54461	8102	301	3	137	6.891865e+18	GALAXY	0.116123	6121	56187	842
5	1.237680e+18	340.995121	20.589476	23.48827	23.33776	21.32195	20.25615	19.54544	8102	301	3	110	5.658977e+18	QSO	1.424659	5026	55855	741

In [6]:
```python
df.info
```

Out[6]:
```
<bound method DataFrame.info of              obj_ID       alpha      delta         u         g         r  \
0      1.237661e+18  135.689107  32.494632  23.87882  22.27530  20.39501
1      1.237665e+18  144.826101  31.274185  24.77759  22.83188  22.58444
2      1.237661e+18  142.188790  35.582444  25.26307  22.66389  20.60976
3      1.237663e+18  338.741038  -0.402828  22.13682  23.77656  21.61162
4      1.237680e+18  345.282593  21.183866  19.43718  17.58028  16.49747
...             ...         ...        ...       ...       ...       ...
99995  1.237679e+18   39.620709  -2.594074  22.16759  22.97586  21.90404
99996  1.237679e+18   29.493819  19.798874  22.69118  22.38628  20.45003
99997  1.237668e+18  224.587407  15.700707  21.16916  19.26997  18.20428
99998  1.237661e+18  212.268621  46.660365  25.35039  21.63757  19.91386
99999  1.237661e+18  196.896053  49.464643  22.62171  21.79745  20.60115

              i         z  run_ID  rerun_ID  cam_col  field_ID   spec_obj_ID  \
0      19.16573  18.79371    3606       301        2        79  6.543777e+18
1      21.16812  21.61427    4518       301        5       119  1.176014e+19
2      19.34857  18.94827    3606       301        2       120  5.152200e+18
3      20.50454  19.25010    4192       301        3       214  1.030107e+19
4      15.97711  15.54461    8102       301        3       137  6.891865e+18
...         ...       ...     ...       ...      ...       ...           ...
99995  21.30548  20.73569    7778       301        2       581  1.055431e+19
99996  19.75759  19.41526    7917       301        1       289  8.586351e+18
99997  17.69034  17.35221    5314       301        4       308  3.112008e+18
99998  19.07254  18.62482    3650       301        4       131  7.601080e+18
99999  20.00959  19.28075    3650       301        4        60  8.343152e+18

        class  redshift  plate    MJD  fiber_ID
0      GALAXY  0.634794   5812  56354       171
1      GALAXY  0.779136  10445  58158       427
2      GALAXY  0.644195   4576  55592       299
3      GALAXY  0.932346   9149  58039       775
4      GALAXY  0.116123   6121  56187       842
...       ...       ...    ...    ...       ...
99995  GALAXY  0.000000   9374  57749       438
99996  GALAXY  0.404895   7626  56934       866
99997  GALAXY  0.143366   2764  54535        74
99998  GALAXY  0.455040   6751  56368       470
99999  GALAXY  0.542944   7410  57104       851

[100000 rows x 18 columns]>
```

In [7]:
```python
df.describe()
```

Out[7]:

	obj_ID	alpha	delta	u	g	r	i	z	run_ID	rerun_ID	cam_col	field_ID	spec_obj_ID
count	1.000000e+05	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.0	100000.000000	100000.000000	1.000000e+05
mean	1.237665e+18	177.629117	24.135305	21.980468	20.531387	19.645762	19.084854	18.668810	4481.366060	301.0	3.511610	186.130520	5.783882e+18
std	8.438560e+12	96.502241	19.644665	31.769291	31.750292	1.854760	1.757895	31.728152	1964.764593	0.0	1.586912	149.011073	3.324016e+18
min	1.237646e+18	0.005528	-18.785328	-9999.000000	-9999.000000	9.822070	9.469903	-9999.000000	109.000000	301.0	1.000000	11.000000	2.995191e+17
25%	1.237659e+18	127.518222	5.146771	20.352353	18.965230	18.135828	17.732285	17.460677	3187.000000	301.0	2.000000	82.000000	2.844138e+18
50%	1.237663e+18	180.900700	23.645922	22.179135	21.099835	20.125290	19.405145	19.004595	4188.000000	301.0	4.000000	146.000000	5.614883e+18
75%	1.237668e+18	233.895005	39.901550	23.687440	22.123767	21.044785	20.396495	19.921120	5326.000000	301.0	5.000000	241.000000	8.332144e+18
max	1.237681e+18	359.999810	83.000519	32.781390	31.602240	29.571860	32.141470	29.383740	8162.000000	301.0	6.000000	989.000000	1.412694e+19

In [8]:
```python
df.columns
```

Out[8]:
```
Index(['obj_ID', 'alpha', 'delta', 'u', 'g', 'r', 'i', 'z', 'run_ID',
       'rerun_ID', 'cam_col', 'field_ID', 'spec_obj_ID', 'class', 'redshift',
       'plate', 'MJD', 'fiber_ID'],
      dtype='object')
```

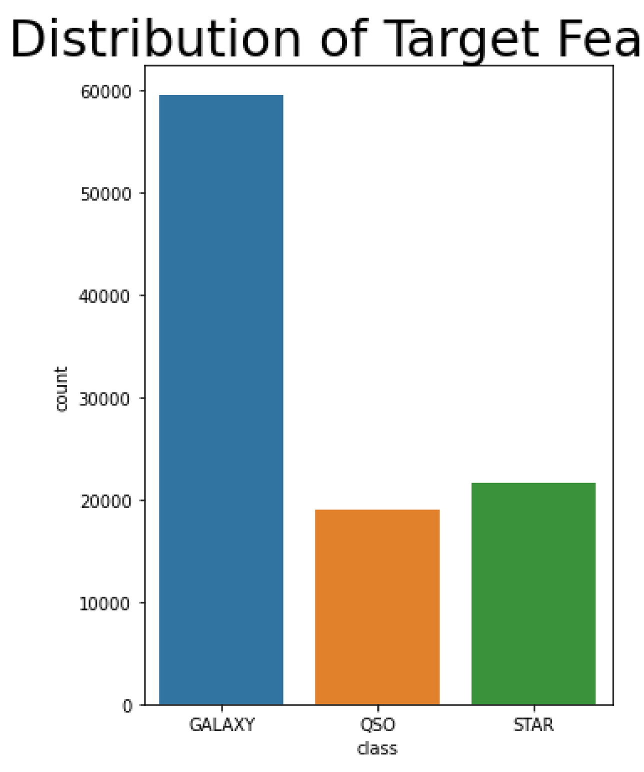
In [9]:
```python
data=df[['alpha', 'delta', 'u', 'g', 'r', 'i', 'z','class','redshift']]
data.head()
```

Out[9]:

	alpha	delta	u	g	r	i	z	class	redshift
0	135.689107	32.494632	23.87882	22.27530	20.39501	19.16573	18.79371	GALAXY	0.634794
1	144.826101	31.274185	24.77759	22.83188	22.58444	21.16812	21.61427	GALAXY	0.779136
2	142.188790	35.582444	25.26307	22.66389	20.60976	19.34857	18.94827	GALAXY	0.644195
3	338.741038	-0.402828	22.13682	23.77656	21.61162	20.50454	19.25010	GALAXY	0.932346
4	345.282593	21.183866	19.43718	17.58028	16.49747	15.97711	15.54461	GALAXY	0.116123

In [10]:
```python
galaxy = data[data['class']=='GALAXY']
star = data[data['class']=='STAR']
qso =data[data['class']=='QSO']


#distribution of class
plt.figure(figsize=(5,7))
sb.countplot(data['class']);
plt.title("Distribution of Target Feature",{'fontsize':30});
```



In [11]:
```python
x=data.drop(['class'],axis='columns')
y=data['class']
```

In [12]:
```python
#LABEL ENCODING(CATEGORICAL DATA)
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

In [13]:
```python
#TRAIN AND TEST DATA
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

In [14]:
```python
#SCALING OF DATA

from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train = scalar.fit_transform(X_train)
X_test  = scalar.transform(X_test)
x=scalar.fit_transform(x)
```

MODEL FITTING

In [21]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
models = {
    KNeighborsClassifier(n_neighbors=3):'K-Neighbors Classifier',
    SVC():"Support Vector Machine",
    RandomForestClassifier():'Random Forest Classifier',
    LogisticRegression():'Logistic Regression',
    GaussianNB():'Naive Bayes'

}
for m in models.keys():
    m.fit(X_train,y_train)
for model,name in models.items():
     print(f"Accuracy Score for {name} is : ",(model.score(X_test,y_test))*100,"%")
```

```
Accuracy Score for K-Neighbors Classifier is :  94.31 %
Accuracy Score for Support Vector Machine is :  95.88 %
Accuracy Score for Random Forest Classifier is :  97.75500000000001 %
Accuracy Score for Logistic Regression is :  95.455 %
Accuracy Score for Naive Bayes is :  74.535 %
```

In [22]:
```python
#PREDICTING ,ACCURACY SCORE BY THE HELP OF CLASSIFICATION REPORT

from sklearn.metrics import classification_report
for model,name in models.items():
    y_pred = model.predict(X_test)
    print(f"Classification Report for : {name}")
    print(classification_report(y_test,y_pred))
```

```
Classification Report for : K-Neighbors Classifier
              precision    recall  f1-score   support

           0       0.95      0.96      0.95     11901
           1       0.95      0.89      0.92      3827
           2       0.92      0.95      0.94      4272

    accuracy                           0.94     20000
   macro avg       0.94      0.93      0.94     20000
weighted avg       0.94      0.94      0.94     20000

Classification Report for : Support Vector Machine
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     11901
           1       0.97      0.89      0.93      3827
           2       0.94      1.00      0.97      4272

    accuracy                           0.96     20000
   macro avg       0.96      0.95      0.95     20000
weighted avg       0.96      0.96      0.96     20000

Classification Report for : Random Forest Classifier
              precision    recall  f1-score   support

           0       0.98      0.99      0.98     11901
           1       0.96      0.93      0.94      3827
           2       0.99      1.00      1.00      4272

    accuracy                           0.98     20000
   macro avg       0.98      0.97      0.97     20000
weighted avg       0.98      0.98      0.98     20000

Classification Report for : Logistic Regression
              precision    recall  f1-score   support

           0       0.96      0.97      0.96     11901
           1       0.94      0.87      0.91      3827
           2       0.95      1.00      0.98      4272

    accuracy                           0.95     20000
   macro avg       0.95      0.94      0.95     20000
weighted avg       0.95      0.95      0.95     20000

Classification Report for : Naive Bayes
              precision    recall  f1-score   support

           0       0.76      0.92      0.83     11901
           1       0.68      0.88      0.77      3827
           2       0.99      0.15      0.26      4272

    accuracy                           0.75     20000
   macro avg       0.81      0.65      0.62     20000
weighted avg       0.79      0.75      0.70     20000
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: