In [109…

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
import warnings
warnings.filterwarnings('ignore')
matplotlib.style.use('ggplot')
```

In [110…

```python
data = pd.read_csv('Skyserver_SQL2_27_2018 6_51_39 PM.csv')
```

In [111…

```python
data.head()
```

Out[111…

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	class	re
0	1.237650e+18	183.531326	0.089693	19.47406	17.04240	15.94699	15.50342	15.22531	752	301	4	267	3.722360e+18	STAR	-0.0
1	1.237650e+18	183.598371	0.135285	18.66280	17.21449	16.67637	16.48922	16.39150	752	301	4	267	3.638140e+17	STAR	-0.0
2	1.237650e+18	183.680207	0.126185	19.38298	18.19169	17.47428	17.08732	16.80125	752	301	4	268	3.232740e+17	GALAXY	0.1
3	1.237650e+18	183.870529	0.049911	17.76536	16.60272	16.16116	15.98233	15.90438	752	301	4	269	3.722370e+18	STAR	-0.0
4	1.237650e+18	183.883288	0.102557	17.55025	16.26342	16.43869	16.55492	16.61326	752	301	4	269	3.722370e+18	STAR	0.0

In [112…

```python
data.tail()
```

Out[112…

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	class
9995	1.237650e+18	131.316413	51.539547	18.81777	17.47053	16.91508	16.68305	16.50570	1345	301	3	161	5.033450e+17	GALAXY
9996	1.237650e+18	131.306083	51.671341	18.27255	17.43849	17.07692	16.71661	16.69897	1345	301	3	162	5.033400e+17	GALAXY
9997	1.237650e+18	131.552562	51.666986	18.75818	17.77784	17.51872	17.43302	17.42048	1345	301	3	162	8.222620e+18	STAR
9998	1.237650e+18	131.477151	51.753068	18.88287	17.91068	17.53152	17.36284	17.13988	1345	301	3	163	5.033400e+17	GALAXY
9999	1.237650e+18	131.665012	51.805307	19.27586	17.37829	16.30542	15.83548	15.50588	1345	301	3	163	5.033410e+17	GALAXY

In [113…

```python
data.columns
```

Out[113…

```
Index(['objid', 'ra', 'dec', 'u', 'g', 'r', 'i', 'z', 'run', 'rerun', 'camcol',
       'field', 'specobjid', 'class', 'redshift', 'plate', 'mjd', 'fiberid'],
      dtype='object')
```

In [114…

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 18 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   objid      10000 non-null  float64
 1   ra         10000 non-null  float64
 2   dec        10000 non-null  float64
 3   u          10000 non-null  float64
 4   g          10000 non-null  float64
 5   r          10000 non-null  float64
 6   i          10000 non-null  float64
 7   z          10000 non-null  float64
 8   run        10000 non-null  int64
 9   rerun      10000 non-null  int64
 10  camcol     10000 non-null  int64
 11  field      10000 non-null  int64
 12  specobjid  10000 non-null  float64
 13  class      10000 non-null  object
 14  redshift   10000 non-null  float64
 15  plate      10000 non-null  int64
 16  mjd        10000 non-null  int64
 17  fiberid    10000 non-null  int64
dtypes: float64(10), int64(7), object(1)
memory usage: 1.4+ MB
```

In [115…

```python
data.describe()
```

Out[115…

	objid	ra	dec	u	g	r	i	z	run	rerun
count	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.0
mean	1.237650e+18	175.529987	14.836148	18.619355	17.371931	16.840963	16.583579	16.422833	981.034800	301.0
std	1.577039e+05	47.783439	25.212207	0.828656	0.945457	1.067764	1.141805	1.203188	273.305024	0.0
min	1.237650e+18	8.235100	-5.382632	12.988970	12.799550	12.431600	11.947210	11.610410	308.000000	301.0
25%	1.237650e+18	157.370946	-0.539035	18.178035	16.815100	16.173333	15.853705	15.618285	752.000000	301.0
50%	1.237650e+18	180.394514	0.404166	18.853095	17.495135	16.858770	16.554985	16.389945	756.000000	301.0
75%	1.237650e+18	201.547279	35.649397	19.259232	18.010145	17.512675	17.258550	17.141447	1331.000000	301.0
max	1.237650e+18	260.884382	68.542265	19.599900	19.918970	24.802040	28.179630	22.833060	1412.000000	301.0

In [116…

```python
data.isnull().sum()
```

Out[116…

```
objid        0
ra           0
dec          0
u            0
g            0
r            0
i            0
z            0
run          0
rerun        0
camcol       0
field        0
specobjid    0
class        0
redshift     0
plate        0
mjd          0
fiberid      0
dtype: int64
```

In [117…

```python
data.shape
```

Out[117…

```
(10000, 18)
```

In [118…

```python
dataframe = pd.DataFrame({'Dtype':data.dtypes,'unique':data.nunique(),'duplicated':data.duplicated().sum()})
```

In [119…

```python
dataframe
```

Out[119…

	Dtype	unique	duplicated
objid	float64	1	0
ra	float64	10000	0
dec	float64	10000	0
u	float64	9730	0
g	float64	9817	0
r	float64	9852	0
i	float64	9890	0
z	float64	9896	0
run	int64	23	0
rerun	int64	1	0
camcol	int64	6	0
field	int64	703	0
specobjid	float64	6349	0
class	object	3	0
redshift	float64	9637	0
plate	int64	487	0
mjd	int64	355	0
fiberid	int64	892	0

In [120…

```python
data.head()
```

Out[120…

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	class	re
0	1.237650e+18	183.531326	0.089693	19.47406	17.04240	15.94699	15.50342	15.22531	752	301	4	267	3.722360e+18	STAR	-0.0
1	1.237650e+18	183.598371	0.135285	18.66280	17.21449	16.67637	16.48922	16.39150	752	301	4	267	3.638140e+17	STAR	-0.0
2	1.237650e+18	183.680207	0.126185	19.38298	18.19169	17.47428	17.08732	16.80125	752	301	4	268	3.232740e+17	GALAXY	0.1
3	1.237650e+18	183.870529	0.049911	17.76536	16.60272	16.16116	15.98233	15.90438	752	301	4	269	3.722370e+18	STAR	-0.0
4	1.237650e+18	183.883288	0.102557	17.55025	16.26342	16.43869	16.55492	16.61326	752	301	4	269	3.722370e+18	STAR	0.0

In [121…

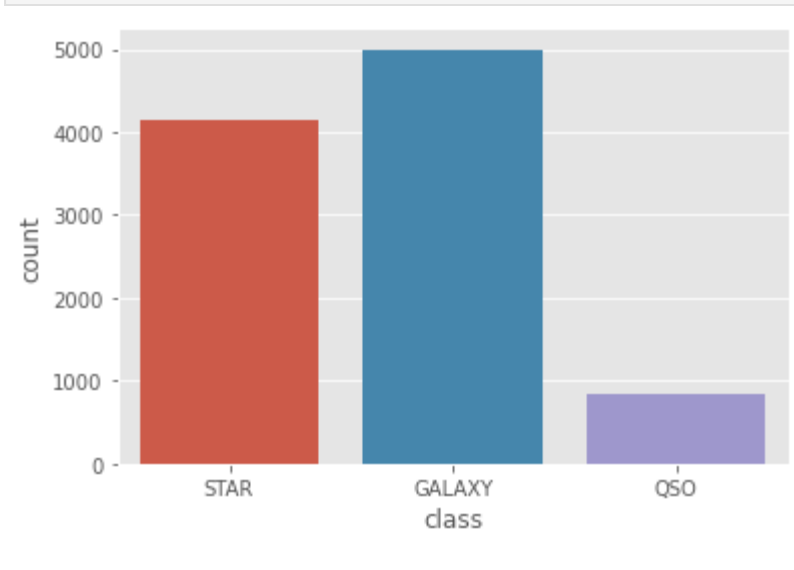
```python
data['class'].value_counts()
```

Out[121…

```
GALAXY    4998
STAR      4152
QSO        850
Name: class, dtype: int64
```

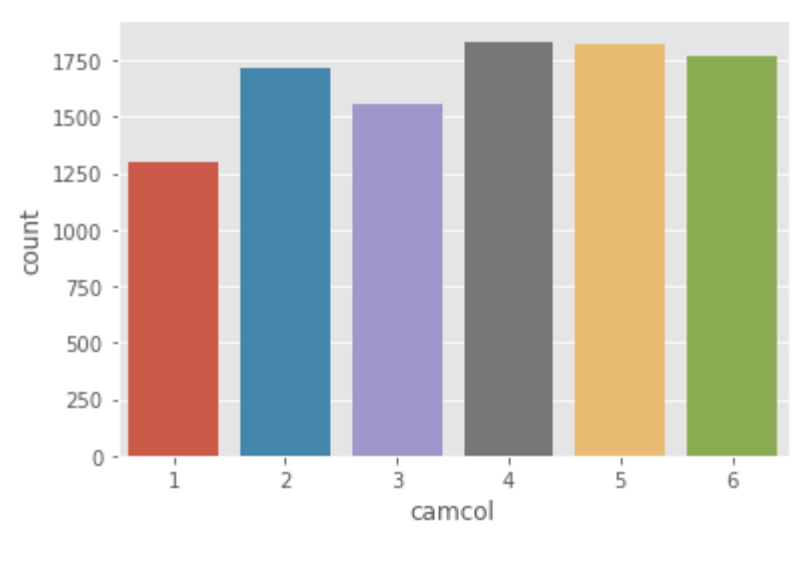
In [122…

```python
sns.countplot(data['class'])
plt.show()
```



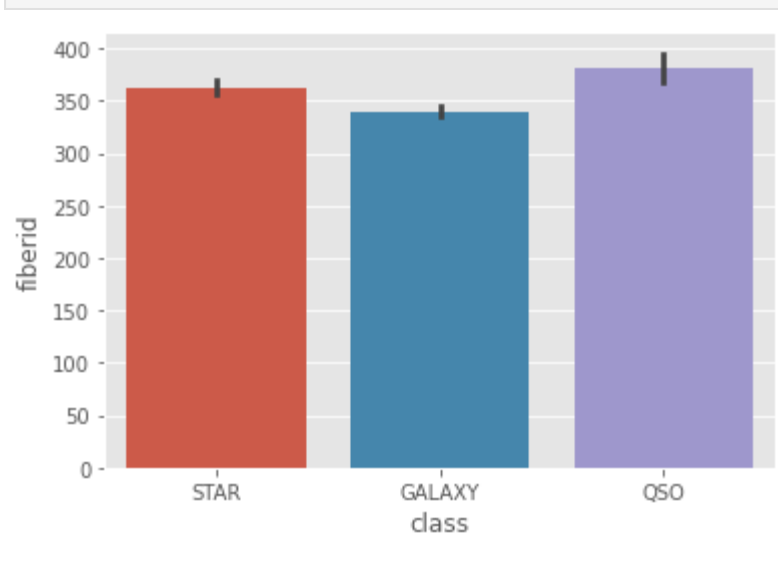
In [123…

```python
sns.countplot(data['camcol'])
plt.show()
```



In [124…

```python
sns.barplot(data['class'],data['fiberid'])
plt.show()
```



In [125…

```python
data1 = data.drop(columns=['class'])
```

In [126…

```python
len(data1.columns)
```

Out[126…

```
17
```

In [127…

```python
data.head()
```

Out[127…

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	class	re
0	1.237650e+18	183.531326	0.089693	19.47406	17.04240	15.94699	15.50342	15.22531	752	301	4	267	3.722360e+18	STAR	-0.0
1	1.237650e+18	183.598371	0.135285	18.66280	17.21449	16.67637	16.48922	16.39150	752	301	4	267	3.638140e+17	STAR	-0.0
2	1.237650e+18	183.680207	0.126185	19.38298	18.19169	17.47428	17.08732	16.80125	752	301	4	268	3.232740e+17	GALAXY	0.1
3	1.237650e+18	183.870529	0.049911	17.76536	16.60272	16.16116	15.98233	15.90438	752	301	4	269	3.722370e+18	STAR	-0.0
4	1.237650e+18	183.883288	0.102557	17.55025	16.26342	16.43869	16.55492	16.61326	752	301	4	269	3.722370e+18	STAR	0.0

In [128…

```python
len(data.columns)
```

Out[128…

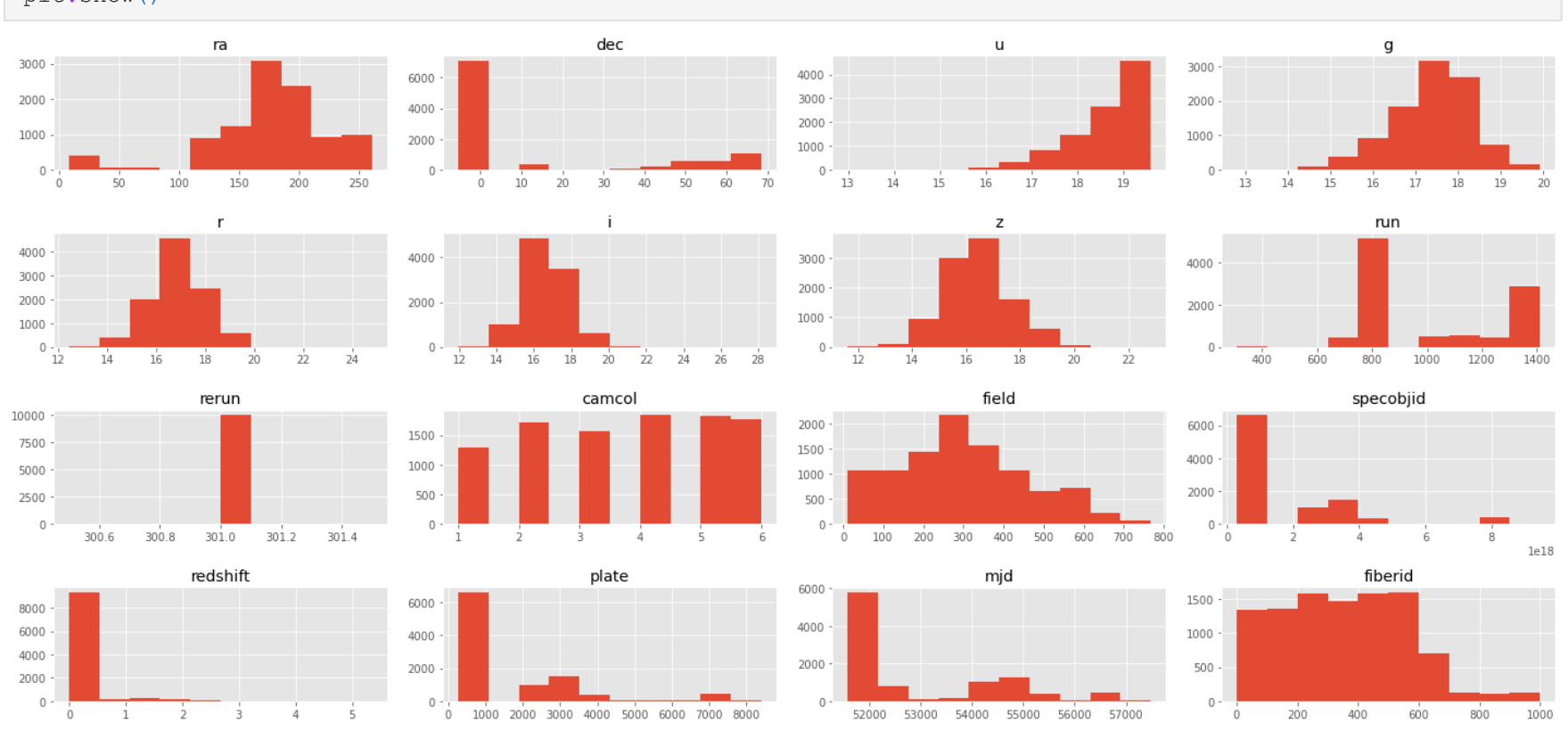
```
18
```

In [129…

```python
data1.drop(columns=['objid'],inplace=True)
```

In [130…

```python
fig,ax = plt.subplots(figsize=(20,9))
data1.hist(ax=ax)
plt.tight_layout()
plt.show()
```



Data-Preprocessing

In [131…

```python
data['class'].value_counts()
```

Out[131…

```
GALAXY    4998
STAR      4152
QSO        850
Name: class, dtype: int64
```

In [132…

```python
from sklearn.preprocessing import LabelEncoder
```

In [133…

```python
lebel = LabelEncoder()
df = lebel.fit_transform(data['class'])
```

In [134…

```python
data['class']=df
```

In [135…

```python
data['class'].value_counts()
```

Out[135…

```
0    4998
2    4152
1     850
Name: class, dtype: int64
```

In [136…

```python
data.head()
```

Out[136…

	objid	ra	dec	u	g	r	i	z	run	rerun	camcol	field	specobjid	class	redsh
0	1.237650e+18	183.531326	0.089693	19.47406	17.04240	15.94699	15.50342	15.22531	752	301	4	267	3.722360e+18	2	-0.0000
1	1.237650e+18	183.598371	0.135285	18.66280	17.21449	16.67637	16.48922	16.39150	752	301	4	267	3.638140e+17	2	-0.0000
2	1.237650e+18	183.680207	0.126185	19.38298	18.19169	17.47428	17.08732	16.80125	752	301	4	268	3.232740e+17	0	0.123
3	1.237650e+18	183.870529	0.049911	17.76536	16.60272	16.16116	15.98233	15.90438	752	301	4	269	3.722370e+18	2	-0.000
4	1.237650e+18	183.883288	0.102557	17.55025	16.26342	16.43869	16.55492	16.61326	752	301	4	269	3.722370e+18	2	0.000

In [137…

```python
data['objid']=data['objid'].astype('int64')
```

In [138…

```python
data['specobjid']= data['objid'].astype('int64')
```

In [139…

```python
corr = abs(data.corr()['class'])
corr
```

Out[139…

```
objid        3.355500e-15
ra           4.321896e-02
dec          5.891815e-02
u            2.690437e-01
g            9.921163e-02
r            4.962769e-02
i            1.467908e-01
z            2.157584e-01
run          7.816172e-02
rerun                 NaN
camcol       6.185920e-03
field        7.712555e-03
specobjid    3.355500e-15
class        1.000000e+00
redshift     7.550956e-02
plate        5.854948e-01
mjd          6.487679e-01
fiberid      5.359261e-02
Name: class, dtype: float64
```

In [140…

```python
good_corr = corr[corr>9.567e-2]
```

In [141…

```python
good_corr
```

Out[141…

```
u        0.269044
g        0.099212
i        0.146791
z        0.215758
class    1.000000
plate    0.585495
mjd      0.648768
Name: class, dtype: float64
```

In [142…

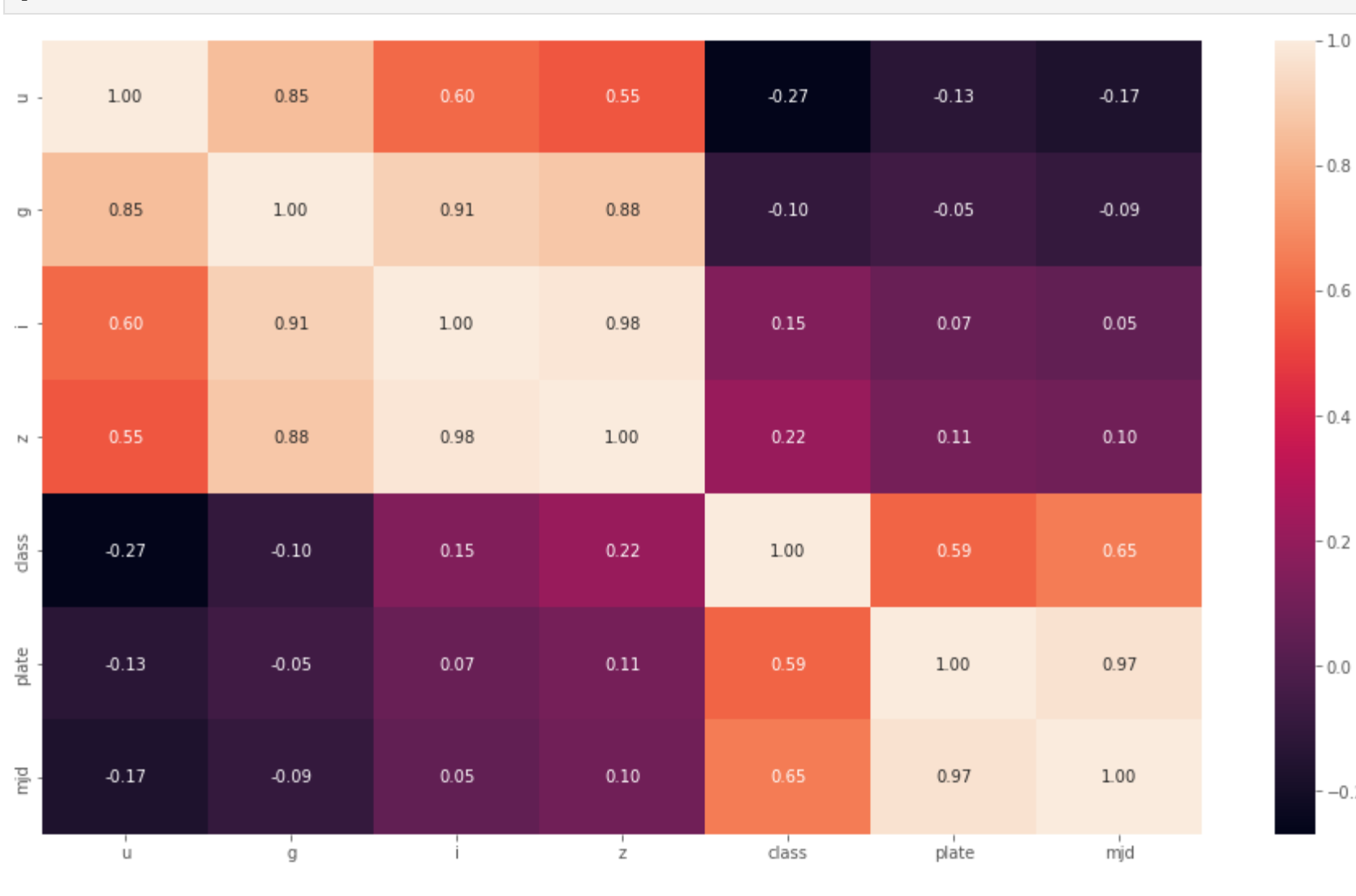
```python
datacorr = data[good_corr.index]
datacorr.head()
```

Out[142…

	u	g	i	z	class	plate	mjd
0	19.47406	17.04240	15.50342	15.22531	2	3306	54922
1	18.66280	17.21449	16.48922	16.39150	2	323	51615
2	19.38298	18.19169	17.08732	16.80125	0	287	52023
3	17.76536	16.60272	15.98233	15.90438	2	3306	54922
4	17.55025	16.26342	16.55492	16.61326	2	3306	54922

In [143…

```python
plt.figure(figsize=(16,9))
sns.heatmap(datacorr.corr(),annot=True,fmt='.2f')
plt.show()
```



In [144…

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

In [145…

```python
x = datacorr.drop(columns=['class'])
y = datacorr['class']
```

In [146…

```python
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,random_state=0,stratify=y)
```

In [147…

```python
x_train.shape,y_train.shape
```

Out[147…

```
((8000, 6), (8000,))
```

In [148…

```python
y_train = y_train.to_numpy()
x_train = x_train.to_numpy()
```

In [149…

```python
print('y_train having 1 :-',len(y_train[y_train==1]))
print('y_train having o :-',len(y_train[y_train==0]))
```

```
y_train having 1 :- 680
y_train having o :- 3998
```

In [150…

```python
from imblearn.over_sampling import SMOTE
```

In [151…

```python
sm = SMOTE(sampling_strategy='minority',random_state=1,k_neighbors=5)
#if it is binary then we can use the 1 but the mupliple clssfication we use the 'minority'.
```

In [152…

```python
x_train_res,y_train_res = sm.fit_resample(x_train,y_train.ravel())
```

In [153…

```python
print('y_train having 1 :-',len(y_train_res[y_train_res==1]))
print('y_train having o :-',len(y_train_res[y_train_res==0]))
```

```
y_train having 1 :- 3998
y_train having o :- 3998
```

In [154…

```python
model = DecisionTreeClassifier(criterion='gini',random_state=42)
model.fit(x_train_res,y_train_res)
y_pred = model.predict(x_test)
```

In [155…

```python
print('Accuarcy :-',accuracy_score(y_test,y_pred))
```

```
Accuarcy :- 0.892
```

In [156…

```python
print('Classification Report :',classification_report(y_test,y_pred))
```

```
Classification Report :               precision    recall  f1-score   support

           0       0.93      0.89      0.91      1000
           1       0.74      0.88      0.80       170
           2       0.88      0.89      0.89       830

    accuracy                           0.89      2000
   macro avg       0.85      0.89      0.87      2000
weighted avg       0.90      0.89      0.89      2000
```

In [157…

```python
analysis = pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
```

In [158…

```python
analysis.head()
```

Out[158…

	Actual	Predicted
9668	2	2
8182	2	2
3749	2	2
7967	2	2
6112	2	2

In [159…

```python
print('Accuracy of our model on the basis of training data :',model.score(x_train_res,y_train_res))
print('Accuracy of our model on the basis of testing data:',model.score(x_test,y_test))
```

```
Accuracy of our model on the basis of training data : 1.0
Accuracy of our model on the basis of testing data: 0.892
```

- Since our able get train 100% but the during the period testing we are getting less accuarcy that means our model is overfitted model.

In [160…

```python
from sklearn.metrics import mean_squared_error
```

In [161…

```python
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[161…

```
0.5665686189686118
```

In [162…

```python
np.std(y_test)
```

Out[162…

```
0.9527722707971635
```

Here we can see that our model is giving less error while modeling if you compare with base model error.

RandomSearchCv

In [163…

```python
from sklearn.model_selection import RandomizedSearchCV,cross_val_score
```

In [164…

```python
param_distributions={'criterion':['gini','entropy'],'min_samples_leaf':np.arange(1,10,1),'max_depth':np.arange
```

In [165…

```python
model_Randgrid = RandomizedSearchCV(model,param_distributions={'criterion':['gini','entropy'],'min_samples_leaf
model_Randgrid.fit(x_train_res,y_train_res)
```

Out[165…

```
RandomizedSearchCV(cv=10, estimator=DecisionTreeClassifier(random_state=42),
                   param_distributions={'ccp_alpha': array([0.  , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.0
8, 0.09, 0.1 ,
       0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
       0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
       0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
       0.44, 0.45, 0.46, 0.47, 0...
       0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,
       0.66, 0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76,
       0.77, 0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87,
       0.88, 0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98,
       0.99]),
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': array([10, 20, 30, 40, 50, 60, 70, 80, 90]),
                                        'min_samples_leaf': array([1, 2, 3, 4, 5, 6, 7, 8, 9])},
                   scoring='accuracy')
```

In [166…

```python
model_Randgrid.best_estimator_
```

Out[166…

```
DecisionTreeClassifier(ccp_alpha=0.05, criterion='entropy', max_depth=70,
                       min_samples_leaf=3, random_state=42)
```

In [167…

```python
model_Randgrid.get_params
```

Out[167…

```
<bound method BaseEstimator.get_params of RandomizedSearchCV(cv=10, estimator=DecisionTreeClassifier(random_sta
te=42),
                   param_distributions={'ccp_alpha': array([0.  , 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.0
8, 0.09, 0.1 ,
       0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
       0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
       0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
       0.44, 0.45, 0.46, 0.47, 0...
       0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,
       0.66, 0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76,
       0.77, 0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87,
       0.88, 0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98,
       0.99]),
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': array([10, 20, 30, 40, 50, 60, 70, 80, 90]),
                                        'min_samples_leaf': array([1, 2, 3, 4, 5, 6, 7, 8, 9])},
                   scoring='accuracy')>
```

In [168…

```python
model_cf = DecisionTreeClassifier(criterion='entropy',max_depth=10,random_state=42,ccp_alpha=0.13)
model_cf.fit(x_train_res,y_train_res)
y_pred = model.predict(x_test)
```

In [169…

```python
print(classification_report(y_test,y_pred))
```

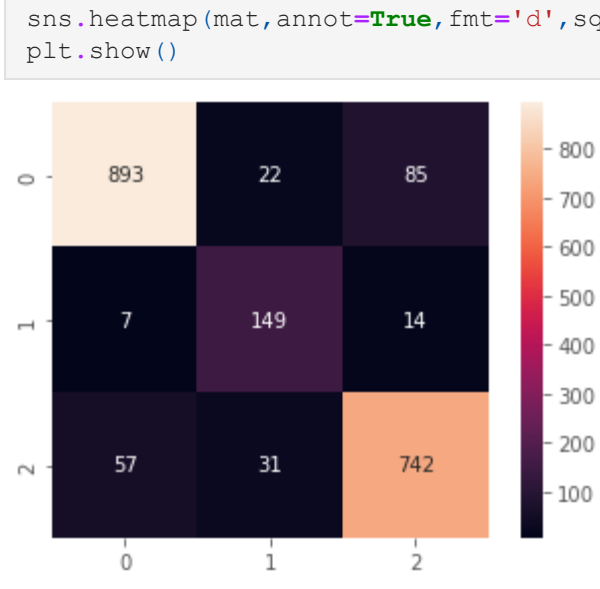
```
              precision    recall  f1-score   support

           0       0.93      0.89      0.91      1000
           1       0.74      0.88      0.80       170
           2       0.88      0.89      0.89       830

    accuracy                           0.89      2000
   macro avg       0.85      0.89      0.87      2000
weighted avg       0.90      0.89      0.89      2000
```

In [175…

```python
mat=confusion_matrix(y_test,y_pred)
```

In [176…

```python
sns.heatmap(mat,annot=True,fmt='d',square=True)
plt.show()
```



In [170…

```python
print('Accuracy of our model on the basis of training data :',model_cf.score(x_train_res,y_train_res))
print('Accuracy of our model on the basis of testing data:',model_cf.score(x_test,y_test))
```

```
Accuracy of our model on the basis of training data : 0.8416681392472168
Accuracy of our model on the basis of testing data: 0.7995
```

In [171…

```python
val_score = cross_val_score(model_cf,x_train_res,y_train_res,cv=15)
val_score
```

Out[171…

```
array([0.84370861, 0.82119205, 0.82649007, 0.83046358, 0.85827815,
       0.85033113, 0.84370861, 0.83311258, 0.85411141, 0.85013263,
       0.83554377, 0.83819629, 0.83421751, 0.84350133, 0.84350133])
```

In [172…

```python
print('variance',np.var(val_score))
print('std',np.std(val_score))
print('mean',np.mean(val_score))
```

```
variance 0.00010153528969575614
std 0.010076472085792534
mean 0.84043260081625
```

From we can conclude the our model is now good for test and train data and as well if you can see the cross validation giving us the low variance and std inside the result so from that we can that our model is good for production level.

In [173…

```python
RMSE = mean_squared_error(y_test,y_pred)
print('Root means Squared error :-',np.std(RMSE))
base_error = np.mean(y_test)
print('base error :-',base_error)
```

```
Root means Squared error :- 0.0
base error :- 0.915
```

From the above we can see that we are getting very less error in the model as compare to base model.

Thanks !!!