

Predicting Solar Flares using Logistic Regression

By-

Sahil Saini

Rachna Chaurasia

25th August 2020

Summary

The document explains a logistic regression model developed to predict the occurrence of solar flares. The model has been programmed using Python. The data has been collected from www.spaceweatherlive.com which is open to the public. Firstly, the reasoning behind selecting this dataset and creating a model for the prediction of flares is given. The data is very uneven as solar flares are rare occasions which provides for a small number of data points for a positive event. Following that, the various steps and techniques for balancing the data such as Synthetic Minority Oversampling Technique (SMOTE), K-Means classification and One-Hot-Encoding are discussed. Finally, the accuracy of the model is reviewed and it is concluded that the logistic regression model is a good way to predict highly imbalanced datasets.

1 Introduction

Our sun is at the centre of the solar system which provides our planet with light and heat in turn being an essential asset for life on Earth. The Sun currently fuses about 600 million tons of hydrogen into helium every second, converting millions of tons of matter into energy constantly as a result.

1.1 Structure of the Sun

The sun various parts of the sun can be classified as follows:

1. **Core** – the innermost 20–25% of the Sun's radius [1], where temperature (energies) and pressure are sufficient for nuclear fusion to occur. Hydrogen fuses into helium and the fusion process releases energy,
2. **Radiative zone** – This is a "radiative zone" in which energy transfer occurs by means of radiation (photons) between about 20–25% of the radius, and 70% of the radius. [2]
3. **Tachocline** – the boundary region between the radiative and convective zones.
4. **Convective zone** – Between about 70% of the Sun's radius and a point close to the visible surface, the Sun is no longer hot enough for radiation to occur but is cool and diffuse enough for convection to occur, and this becomes the primary means of outward heat transfer.
5. **Photosphere** – the deepest part of the Sun which we can directly observe with visible light. [3] Because the Sun is a gaseous object, it does not have a clearly defined surface; its visible parts are usually divided into a 'photosphere' and 'atmosphere'.
6. **Atmosphere** – a gaseous covering surrounding the Sun, comprising the chromosphere, solar transition region, corona and heliosphere. These can be seen when the main part of the Sun is hidden, for example, during a solar eclipse.

1.2 Sunspots and Solar Flares

Sunspots are dark visible spots on the sun which are a result of strong magnetic field lines coming up from within the sun. They are darker because they are significantly cooler than their surroundings. A complex collection of sunspots in an area is called a sunspot region or active region. They are constantly analysed and recorded due to their eruptive threat. [4]

Solar Flares large and sudden eruptions of energy into space due to the entanglement, or reorganization of magnetic field lines. These eruptions release a large amount of radiation which may hit our planet as well. [5]

1.3 Sunspot Region Summary

The Solar Region Summary (SRS), compiled by SWPC, is a daily report of the active solar regions observed during the preceding day. The SRS contains a detailed description of the active regions currently visible on the solar disk. See sample and description below.

The characteristics for each active region are compiled from up to six observatories that report to the SWPC in near-real time. The sunspot counts are typically higher than those reported in non-real time by the Sunspot Index Data Center (SIDC), Brussels, Belgium, and the American Association of Variable Star Observers.

Description: [6]

Nmbr: An SESC region number assigned to a sunspot group during its disk passage.

Location: Sunspot group location, in heliographic degrees latitude and degrees east or west from central meridian, rotated to 2400 UTC.

Lo: Carrington longitude of the group.

Area: Total corrected area of the group in millionths of the solar hemisphere.

Z: Modified Zurich classification of the group.

LL: Longitudinal extent of the group in heliographic degrees.

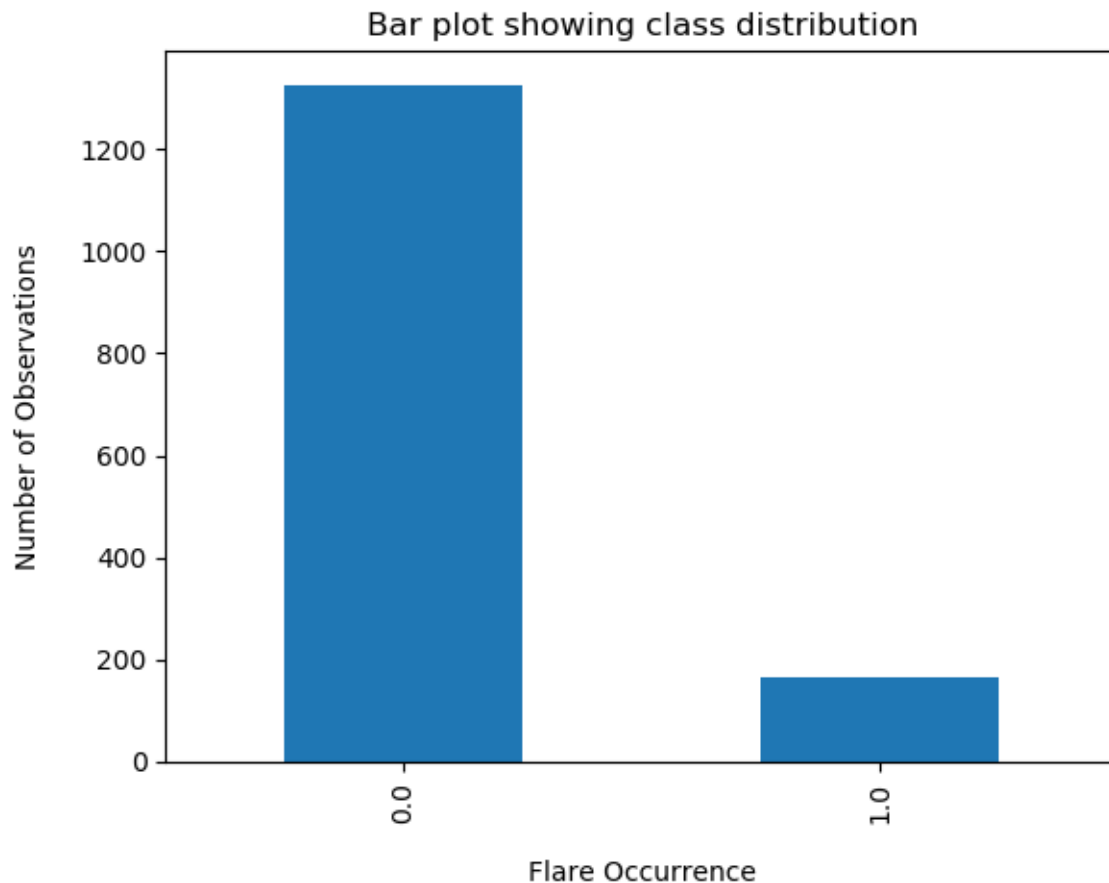
NN: Total number of visible sunspots in the group.

Mag Type: Magnetic classification of the group.

1.4 Characteristics of the Data

The data used here is a modification of the solar region summary (SRS) data. We also have the data of solar weather events prepared by U.S. Dept. of Commerce, NOAA, Space Weather Prediction Center available on the spwc.noaa.gov website. This data allows us to identify the sunspot region and the date and time at which a solar flare occurred.

We know that a solar flare is a rare occurrence, which contributes heavily to the imbalanced data.



Plot A) This plot shows the total number of observations of Non-Occurrence (Zeros) and Occurrences (Ones)

The fitting of this data is very tedious and inaccurate because it is so highly skewed in favour of non-occurrences of solar flares.

1.5 Objective

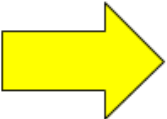
The focus of this project is to treat the imbalanced data with the use of Synthetic Minority Oversampling Technique (SMOTE) to increase the number of observations of flare occurrences and decrease the number of non-occurrences using K-means clustering.

The new data shall then be fitted with a logistic regression and then judged based on Learning curve and Receiver Operating Characteristic (ROC) Curve.

2 One-Hot Encoding

In SRS data, the variables Z: Modified Zurich classification of the sunspot region and Mag Type: Magnetic classification of the sunspot region are categorical variables. These types of variables cause a problem because most machine learning algorithms cannot operate on label data directly and need all the variables to be numerical. “This is mostly a constraint of the efficient implementation of machine learning algorithms rather than hard limitations on the algorithms themselves.” [11]

One-Hot Encoding is very popular and widespread method to turn categorical variables to numeric for computational ease and efficient learning by the algorithm. This technique creates new (binary) columns, indicating the presence of each possible value from the original data.



Color	Red	Yellow	Green
Red	1	0	0
Red	1	0	0
Yellow	0	1	0
Green	0	0	1
Yellow	0	0	1

Image B) shows the categorical variable colour being turned into numerical variables.

After using One-Hot Encoding on Z and Mag Type, the data now had a large number of variables and many of which had very low frequencies. Including all the variables in the model would not be ideal and keep the model very complex as well as low performing.

To deal with this problem, only top appearing variables were taken. Top 50 variables for Z and Top 5 for Mag type were finally included into the solution.

3 Balancing the Data

The unevenness of the data results for very variable results depending on the observations that are randomly chosen for the training and testing datasets. That is why this type of data requires balancing so that the results are much more consistent with a higher accuracy overall.

3.1 SMOTE: Synthetic Minority Over-Sampling Technique

The performance of machine learning algorithms is typically measured using predictive accuracy. However, it is not ideal when the data is imbalanced. [7] Classification using imbalanced data is almost always presents a bias towards the majority class. The problem can be resolved under-sampling of the majority class or oversampling the minority. In most cases, under-sampling yields better results than random oversampling, but SMOTE is a popular technique that improves upon random over-sampling. [8]

In SMOTE, by operating in the 'feature space ', synthetic examples of the minority data are generated. The minority class for the training data is over-sampled by taking each minority class sample and introducing new values along the line segments joining the randomly chosen K nearest neighbours of the minority class. [7]

Here, the solar flare data is heavily biased towards the non-occurrences of solar flares. So, over-sampling the flare occurrence class would be a step in the right direction. After splitting the data into training and testing data sets, the minority class of the training data was over-sampled up-to 400 observations. However, this number is still less than the observations provided by the majority class. The reasoning behind this is that under-sampling of the majority class combined with the over-sampling of the minority class gives a better prediction accuracy as it provides a larger presence for the minority class without a very high degree of over-sampling as suggested by Chawla, Bowyer, Hall & Kegelmeyer in 2002 in their journal article. [7]

So, the next section explains the technique used to under-sample the non-occurrences which is K Means Clustering.

3.2 K-Means Clustering

Clustering is the task of identifying/classifying data into subgroups such that within a subgroup the data values are similar whereas in the data points in different subgroups are different. K-Means clustering is defined as the task of clustering the data into a predefined number of clusters 'K', such that each data point belongs only to one cluster. [9]

The objective that K-Means follows is that of Expectation Maximization with the objective function:

$$J = \sum_{i=1}^m \sum_{k=1}^k w_{ik} ||x^i - \mu_k||^2$$

where $w_{ik}=1$ for data point x^i if it belongs to cluster k ; otherwise, $w_{ik}=0$. Also, μ_k is the centroid of x^i 's cluster

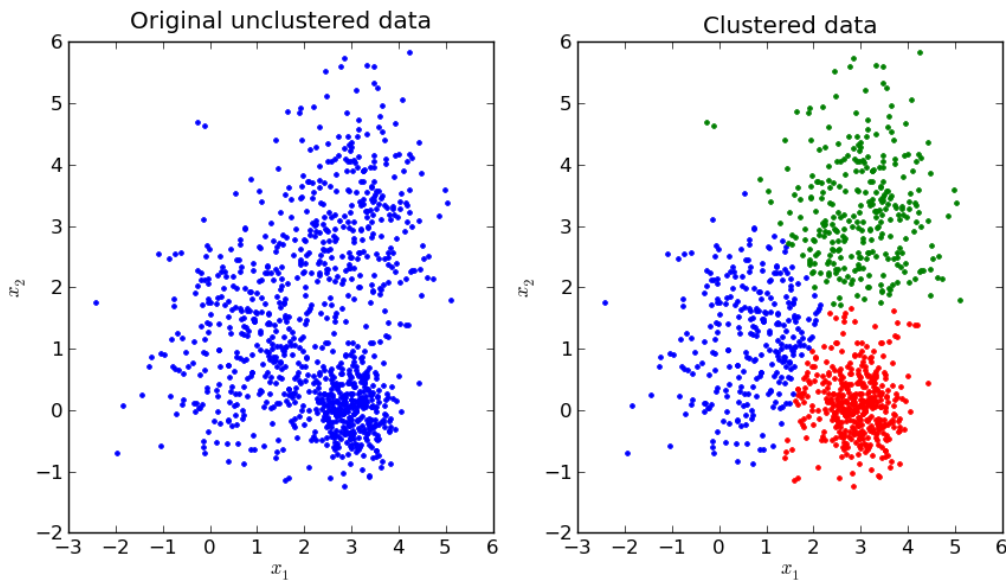
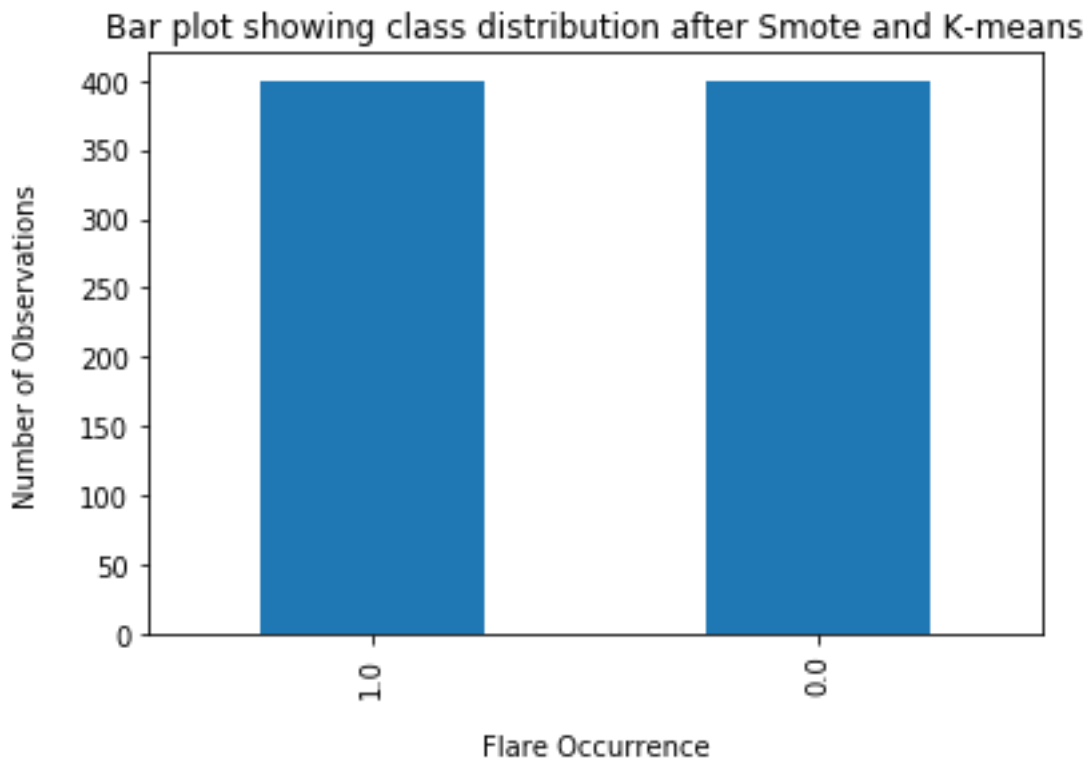


Image A) Shows the comparison between the raw data and the clustered data after using K-means Clustering. (Viswarupan, 2017) [10]

Here, we have used K-Means Clustering to under-sample the majority class 'non-occurrences' of solar flares into 400 data points which is now equal to the 400 data points of 'flare occurrences' in the training data. The new data points of the majority class are the centroids of $K = 400$ clusters.

The resultant class distribution of the training dataset is given below in Plot B,



Plot B) This plot shows the new total number of observations of Non-Occurrence (Zeros) and Occurrences (Ones)

4 Logistic Regression in Machine Learning

After scrutinizing the data and getting the balanced dataset, the next step was to apply the logistic regression algorithm to the data for the prediction of solar flares.

Logistic regression is a classification algorithm, used when the value of the target variable is categorical in nature. Logistic regression is most commonly used when the data in question has binary output, so when it belongs to one class or another, or is either a 0 or 1. Our response variable i.e. whether a flare occurs or not is evidently a binary output and hence logistic regression was a clear choice.

This regression technique uses a sigmoid function/logistic function which has an 'S' shape when plotted. The sigmoid functions pushes the values towards the margins that are 0 and 1. It gives an output 0 if the input is a large negative number and 1 if the input is a large positive number. This is the basis of the classification algorithm that is the logistic regression.

The sigmoid function is as follows:

$$Y = 1 / (1 + e^{-x})$$

5 Results and Conclusion

The training and testing data split used was 60-40. Only the training data was scrutinized and balanced to avoid information leak. Information leak would lead to a better accuracy but only for this data, overall, the accuracy would be very different.

Table 1: Contingency table summarising logistic regression solar flare predictions

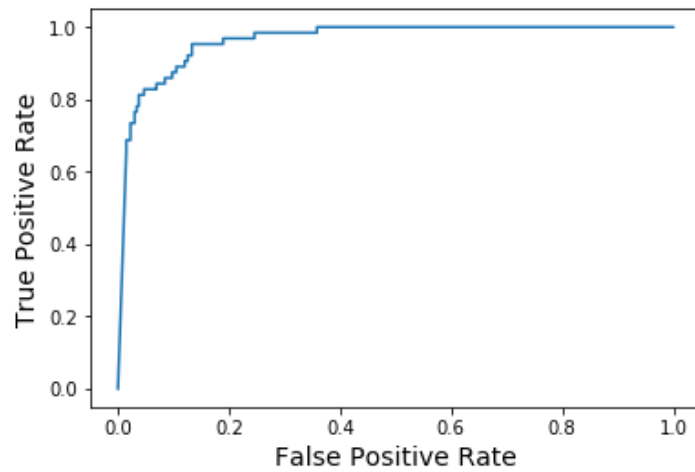
		Forecast		
		Flare	No Flare	Total
Observed	Flare	60	4	64
	No Flare	7	526	533
Total		67	530	597

	precision	recall	f1-score	support
0.0	0.99	0.99	0.99	533
1.0	0.90	0.94	0.92	64
accuracy			0.98	597
macro avg	0.94	0.96	0.95	597
weighted avg	0.98	0.98	0.98	597

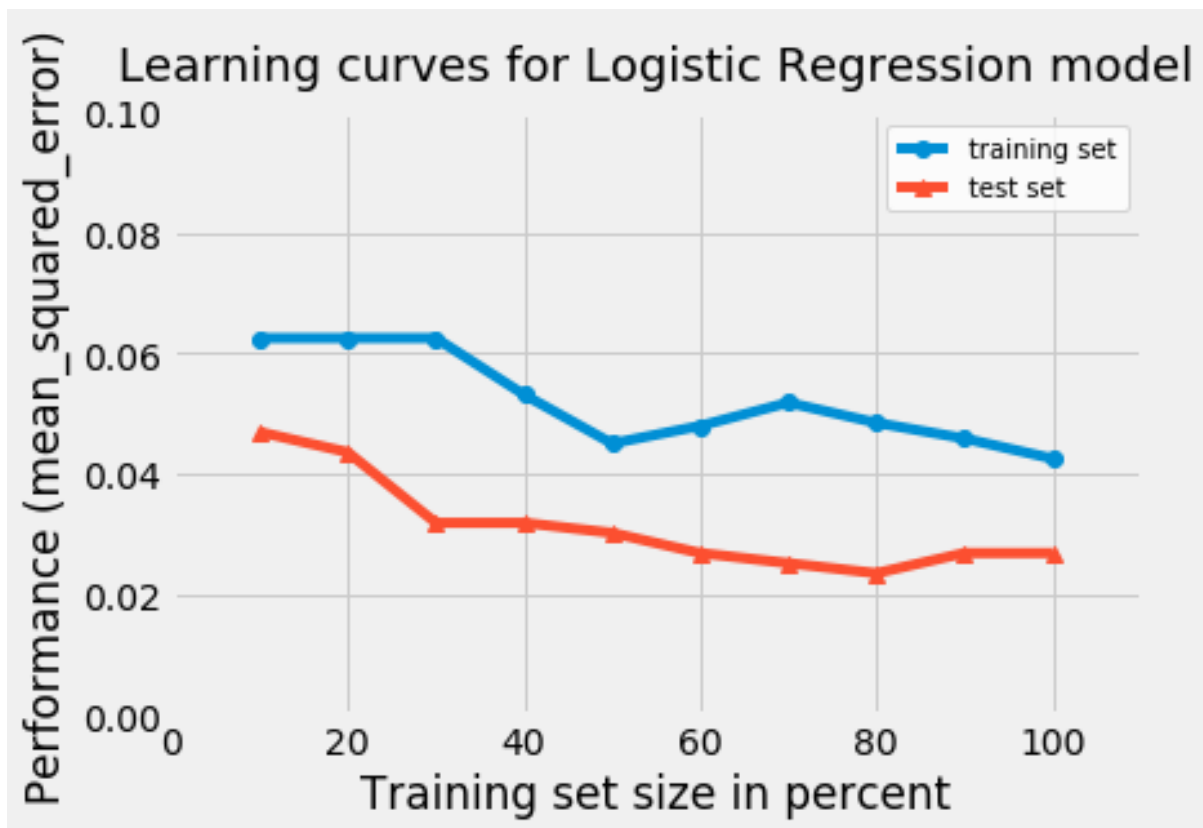
In this Project, Our main aim was to give an efficient model that is able to predict accurately whether a flare will be emitted or not. The Logistic model is giving the accuracy for class 1 of 92% and for class 0 of 99% on the test data.

Plot 3: ROC curve for Logistic Regression

Receiver Operating Characteristic (ROC) Curve for Logistic Regression



Plot 4: Learning Curve for Logistic Regression



Appendix 1: Code

```
from mlxtend.plotting import plot_learning_curves
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt; plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from sklearn.decomposition import PCA

data1= pd.read_csv('solarflare.csv')
data2=pd.read_csv('SolarFlares1.csv')
data1=data1[['Area', 'Z', 'NN', 'Mag_Type', 'Growth', 'class']]
data2=data2[['Area', 'Z', 'NN', 'Mag_Type', 'Growth', 'class']]

data=pd.concat([data1,data2],ignore_index=True)
data=shuffle(data,random_state=1) #shuffling of the observations
data.reset_index(inplace=True,drop=True)
print(data)

#finding number of missing values
data=data.dropna(how='any') #addressing missing values
data.Z.value_counts().sort_values(ascending=False) #for getting
how many times a class occurs in 'Z'
```

```

data.shape

top_50=[x for x in
data.Z.value_counts().sort_values(ascending=False).head(50).index]
top_3=[x for x in
data.Growth.value_counts().sort_values(ascending=False).head(3).index]
top_5=[x for x in
data.Mag_Type.value_counts().sort_values(ascending=False).head(5).index
]
for label in top_20:
    data[label]=np.where(data['Z']==label,1,0)
a=data[top_20]
for label in top_3:
    data[label]=np.where(data['Growth']==label,1,0)
b=data[top_3]
for label in top_5:
    data[label]=np.where(data['Mag_Type']==label,1,0)
c=data[top_5]
d=pd.concat([a,b,c],axis=1)
e=data[['Area','NN','class']]
finaldata=pd.concat([d,e],axis=1)
finaldata.shape

```

```

def clean_dataset(df):
    assert isinstance(df, pd.DataFrame), "df needs to be a
pd.DataFrame"
    df.dropna(inplace=True)
    indices_to_keep = ~df.isin([np.nan, np.inf, -np.inf]).any(1)
    return df[indices_to_keep].astype(np.float64)
finaldata=clean_dataset(finaldata)
finaldata['class'].value_counts().plot(kind='bar')
plt.title("Bar plot showing class distribution before Smote(over
sampling)")
plt.xlabel("Class", labelpad=14)
plt.ylabel("Flare occurance", labelpad=14)

```

```

train_set, test_set=
train_test_split(finaldata,test_size=0.4,random_state=15) #splitting
of data into test and train

```

```

dataset = {}
#grouping the dataset with respect to the classes
by_class = train_set.groupby('class')
for groups, values in by_class:
    dataset[groups] = values

```

```

X1= dataset[0]      # data of only class 0
X2= dataset[1]      # data of only class 1

```

```

X1=X1.iloc[:,0:60].values

#applying K-Means clustering to the data of class label 0
cluster= KMeans(n_clusters=400, random_state=42)
cluster.fit(X1)
Xtransform= cluster.cluster_centers_      #to get the cenroids
b= np.zeros([400,1],dtype=int)           # creating an array of 0's
Xt=np.hstack((Xtransform,b))
X0=pd.DataFrame(Xt)
X0.columns=X2.columns
underdata=pd.concat([X0,X2],ignore_index=True)
underdata=shuffle(underdata,random_state=1)      #shuffling of the
observations
underdata.reset_index(inplace=True,drop=True)
print(underdata)

underdata['class'].value_counts().plot(kind='bar')

columns=underdata.columns.tolist()
columns=[c for c in columns if c not in ["class"]]
X=underdata[columns]
Y=underdata['class']
import imblearn                        #performing oversampling
oversample = SMOTE()
x,y = oversample.fit_resample(X, Y)
bdata=pd.concat([x,y],axis=1)

bdata['class'].value_counts().plot(kind='bar')
plt.title("Bar plot showing class distribution after Smote(over
sampling)")
plt.xlabel("Class", labelpad=14)
plt.ylabel("Flare occurance", labelpad=14)

Xtrain= bdata.iloc[:,0:60].values
ytrain= bdata.iloc[:,60:61].values
Xtest= test_set.iloc[:,0:60].values
ytest= test_set.iloc[:,60:61].values

#normalize the data
scaler=StandardScaler()
scaler.fit(Xtrain)
Xtrain1= scaler.transform(Xtrain)
Xtest1= scaler.transform(Xtest)

#fitting of logistic regression model
lr= LogisticRegression(solver='lbfgs',random_state=42,max_iter=10000)
lr.fit(Xtrain1,ytrain.ravel())
#predicting y for test data
y_pred= lr.predict(Xtest1)
accuracy= metrics.classification_report(ytest,y_pred)
print(accuracy)
Con_M= metrics.confusion_matrix(ytest,y_pred)

```

```
print(Con_M)
```

```
#Parameters for ROC plot
Prob1= lr.predict_proba(Xtest)[:,-1]
Fp1, tp1, thresholds2 = metrics.roc_curve(ytest, Prob1)
AUC2 = metrics.roc_auc_score(ytest, Prob1)
print(AUC2)
# ROC curve
plt.plot(fp1,tp1)
plt.title('Receiver Operating Characteristic (ROC) and Logistic
Regression',
          fontsize = 18, y = 1.03)
plt.ylabel('True Positive Rate', fontsize = 14)
plt.xlabel('False Positive Rate', fontsize = 14)
plt.show()
```

```
#learning curves
plot_learning_curves(Xtrain, ytrain.ravel(), Xtest, ytest.ravel(), lr,
                     scoring='mean_squared_error')
plt.title('Learning curves for Logistic Regression model', fontsize =
18, y = 1.03)
plt.ylim(0,0.10)
plt.legend()
plt.show()
```