In [19]:
```
#Packages initialisation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error,
```

In [4]:
```
#Wine Dataset load
wine_df=pd.read_csv(r"C:\\Nishant\White_WineQuality_Data.csv")
```

In [6]:
```
#Check column details of the dataset
wine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         4898 non-null   float64
 1   volatile acidity      4898 non-null   float64
 2   citric acid           4898 non-null   float64
 3   residual sugar        4898 non-null   float64
 4   chlorides             4898 non-null   float64
 5   free sulfur dioxide   4898 non-null   float64
 6   total sulfur dioxide  4898 non-null   float64
 7   density               4898 non-null   float64
 8   pH                    4898 non-null   float64
 9   sulphates             4898 non-null   float64
 10  alcohol               4898 non-null   float64
 11  quality               4898 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
```

In [5]:
```
#Check whether any of the columns have null values
wine_df.isna().sum(axis = 0)
```

Out[5]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

In [7]:
```
# Dependent variable and independent variables formation
indp_vrbl = wine_df.drop(['quality'],axis=1)
dep_vrbl = wine_df['quality']
```

In [8]:
```
#Normalizing the independent variables data
normalzd= MinMaxScaler((-1,1))
indp_vrbl_normlzd=normalzd.fit_transform(indp_vrbl)
```

In [9]:
```
#Splitting the variables
X_trn,X_tst,Y_trn,Y_tst = train_test_split(indp_vrbl_normlzd, dep_vrbl ,test_s
```

In [11]:
```
#K- Nearest Neighbour algorithm
import warnings
with warnings.catch_warnings(record=True):
    knn_algrthm = KNeighborsClassifier(n_neighbors = 5, metric = 'euclidean',
    knn_algrthm.fit(X_trn, Y_trn)
    Y_predct1 = knn_algrthm.predict(X_tst)
    print("KNN Algorithm's classification report equals:\n",classification_rep
    print("KNN Algorithm's mean absolute error equals:",mean_absolute_error(Y_
    print("KNN Algorithm's root mean squared error equals:",np.sqrt(mean_squar
    print("KNN Algorithm's r-squared value equals:",r2_score(Y_tst, Y_predct1)
```

```
KNN Algorithm's classification report equals:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.20      0.07      0.10        29
           5       0.54      0.63      0.59       280
           6       0.58      0.62      0.60       450
           7       0.51      0.43      0.47       180
           8       0.38      0.16      0.22        38
           9       0.00      0.00      0.00         1

    accuracy                           0.55       980
   macro avg       0.32      0.27      0.28       980
weighted avg       0.54      0.55      0.54       980

KNN Algorithm's mean absolute error equals: 0.5040816326530613
KNN Algorithm's root mean squared error equals: 0.7915368672682896
KNN Algorithm's r-squared value equals: 0.17873545569317917
```

In [15]:
```
#Support Vector Machine algorithm
import warnings
with warnings.catch_warnings(record=True):
    svm_algrthm = SVC(kernel='linear', C=5, gamma=0.1)
    svm_algrthm.fit(X_trn, Y_trn)
    Y_predct2 = svm_algrthm.predict(X_tst)
    print("Support Vector Machine Algorithm's classification report equals:\n"
    print("Support Vector Machine Algorithm's mean absolute error equals:",mea
    print("Support Vector Machine Algorithm's root mean squared error equals:"
    print("Support Vector Machine Algorithm's r-squared value equals:",r2_scor
```

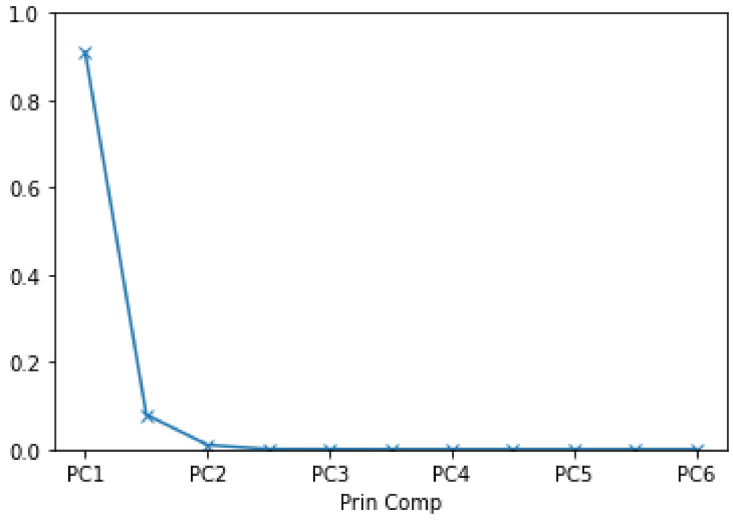
```
Support Vector Machine Algorithm's classification report equals:
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.00      0.00      0.00        29
           5       0.53      0.51      0.52       280
           6       0.51      0.80      0.62       450
           7       0.00      0.00      0.00       180
           8       0.00      0.00      0.00        38
           9       0.00      0.00      0.00         1

    accuracy                           0.51       980
   macro avg       0.15      0.19      0.16       980
weighted avg       0.38      0.51      0.43       980

Support Vector Machine Algorithm's mean absolute error equals: 0.554081632653
0612
Support Vector Machine Algorithm's root mean squared error equals: 0.83726961
49767201
Support Vector Machine Algorithm's r-squared value equals: 0.0810932541713583
8
```

In [17]:
```
#Decision Tree algorithm
import warnings
with warnings.catch_warnings(record=True):
    dtree_algrthm = DecisionTreeClassifier(random_state=5)
    dtree_algrthm.fit(X_trn, Y_trn)
    Y_predct3 = dtree_algrthm.predict(X_tst)
    print("Decision Tree Algorithm's classification report equals:\n",classifi
    print("Decision Tree Algorithm's mean absolute error equals:",mean_absolut
    print("Decision Tree Algorithm's root mean squared error equals:",np.sqrt(
    print("Decision Tree Algorithm's r-squared value equals:",r2_score(Y_tst,
```

```
Decision Tree Algorithm's classification report equals:
               precision    recall  f1-score   support

           3       0.00      0.00      0.00         2
           4       0.31      0.38      0.34        29
           5       0.61      0.64      0.62       280
           6       0.66      0.63      0.65       450
           7       0.57      0.54      0.55       180
           8       0.43      0.53      0.47        38
           9       0.00      0.00      0.00         1

    accuracy                           0.60       980
   macro avg       0.37      0.39      0.38       980
weighted avg       0.61      0.60      0.61       980

Decision Tree Algorithm's mean absolute error equals: 0.5051020408163265
Decision Tree Algorithm's root mean squared error equals: 0.8730663766055727
Decision Tree Algorithm's r-squared value equals: 0.0008393899068481758
```

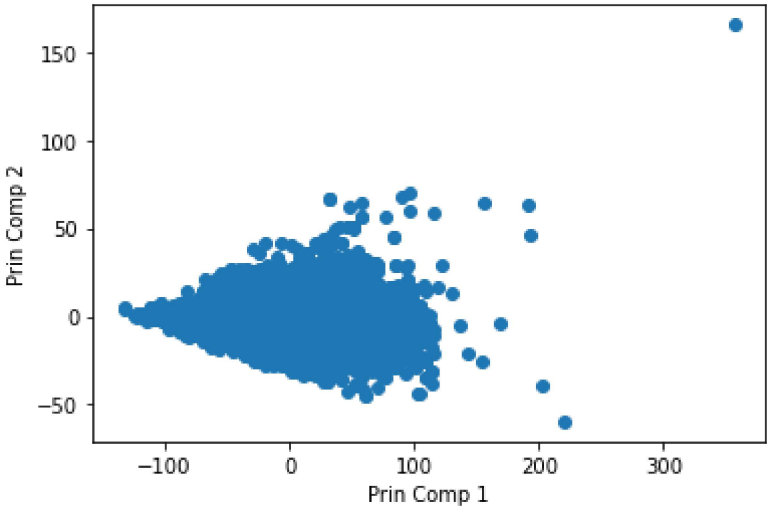
In [21]:
```
#Principal Component Analysis algorithm
prncpal_com_analysis = PCA()
dta = indp_vrbl.values
prncpal_com_analysis.fit(dta)
wine_prncpal_com_analysis = prncpal_com_analysis.transform(dta)
```

In [23]:
```
#Scree plot for obtaining cluster count
import warnings
with warnings.catch_warnings(record=True):
    crd = plt.gca()
    var_exp = prncpal_com_analysis.explained_variance_ratio_
    crd.plot(var_exp, marker='x')
    crd.set_xlabel('Prin Comp')
    crd.set_ylim(0,1.)
    crd.set_xticklabels(["PC{}".format(i) for i in range(8)])
```



In [24]:
```
#PC1 vs PC2 visualisation
crd = plt.gca()
crd.scatter(wine_prncpal_com_analysis[:,0], wine_prncpal_com_analysis[:,1])
crd.set_xlabel('Prin Comp 1')
crd.set_ylabel('Prin Comp 2')
```

Out[24]: Text(0, 0.5, 'Prin Comp 2')



In [25]:
```
#Variance of dataset explained by Principal Components
print("Variance of dataset explained by Principal Components equals->",var_exp
```

```
Variance of dataset explained by Principal Components equals-> [9.09657344e-0
1 7.93338631e-02 1.01542742e-02 5.06004450e-04
 3.23409395e-04 8.72769740e-06 6.72986618e-06 5.39060918e-06
 4.07002123e-06 1.86525322e-07 1.49217279e-10]
```

In [ ]:
```

```