In [108]:
```python
# Here dataset without scaling is used for training the models.
def models(X_train,Y_train):
    #Using GaussianNB method of naïve_bayes class to use Naïve Bayes Algorithm
    gauss = GaussianNB()
    gauss.fit(X_train, Y_train)

    #Using DecisionTreeClassifier of tree class to use Decision Tree Algorithm
    tree = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
    tree.fit(X_train, Y_train)

    #Using RandomForestClassifier method of ensemble class to use Random Forest Classification algorithm
    forest = RandomForestClassifier(max_depth=70, max_features=2, min_samples_leaf=3,
                        min_samples_split=8, n_estimators=300,random_state =22)
    forest.fit(X_train, Y_train)

  #print model accuracy on the training data.

    print('[4]Gaussian Naive Bayes Training Accuracy:', gauss.score(X_train, Y_train))
    print('[5]Decision Tree Classifier Training Accuracy:', tree.score(X_train, Y_train))
    print('[6]Random Forest Classifier Training Accuracy:', forest.score(X_train, Y_train))

    return gauss, tree, forest
```

In [109]:
```python
#Get and train all of the models
model = models(x_train,y_train)
```

```
[4]Gaussian Naive Bayes Training Accuracy: 0.5731039874902267
[5]Decision Tree Classifier Training Accuracy: 1.0
[6]Random Forest Classifier Training Accuracy: 0.8881939014855356
```

In [110]:
```python
# From the above results we can conclude that Random Forest Classifier is giving a better training accuarcy of
# Eventhough Decision Tree Classifier is giving 100% accuaracy we can not rely on it.
# It may be due to over fitting as well.
# Let us evaluate the Random forest model by finding the accuracy score of test data
#plotting confusion matrix, auc curve etc. Then finally let us
# do hyperparameter tuning as well.
```

In [111]:
```python
# Instantiate and fit the RandomForestClassifier
forest = RandomForestClassifier(max_depth=70, max_features=2, min_samples_leaf=3,
                        min_samples_split=8, n_estimators=300,random_state =22)
forest.fit(x_train, y_train)
# Make predictions for the test set
y_pred_rf = forest.predict(x_test)
# View accuracy score
accuracy_score(y_test, y_pred_rf)
```

Out[111]:
```
0.63125
```

In [32]:
```python
# Here we can see that the test accuracy is 63.125% and  training accuracy is88.88% , which means that the mode
# overfitting. We have to do hyperparameter tuning and cross validation to make it better.
```

In [112]:
```python
# View confusion matrix for test data and predictions
confusion_matrix(y_test, y_pred_rf)
```

Out[112]:
```
array([[ 0,  0,  1,  0,  0,  0],
       [ 0,  0,  7,  3,  0,  0],
       [ 0,  0, 97, 33,  0,  0],
       [ 0,  0, 35, 92,  5,  0],
       [ 0,  0,  0, 29, 13,  0],
       [ 0,  0,  0,  2,  3,  0]], dtype=int64)
```

In [113]:
```python
# View the classification report for test data and predictions
print(classification_report(y_test, y_pred_rf))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00        10
           5       0.69      0.75      0.72       130
           6       0.58      0.70      0.63       132
           7       0.62      0.31      0.41        42
           8       0.00      0.00      0.00         5

    accuracy                           0.63       320
   macro avg       0.32      0.29      0.29       320
weighted avg       0.60      0.63      0.61       320
```

GridSearchCV

In [114]:
```python
from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [70,80, 90],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300,1000],
    'criterion': ['gini', 'entropy'],
    'oob_score': [False],
    'random_state': [42],
}
# Create a based model
forest = RandomForestClassifier()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = forest, param_grid = param_grid,
                          cv = 9, n_jobs = 3, verbose = True)
```

In [115]:
```python
grid_search.fit(x_train, y_train)
grid_search.best_params_
```

Out[115]:
```
Fitting 9 folds for each of 432 candidates, totalling 3888 fits
{'bootstrap': True,
 'criterion': 'entropy',
 'max_depth': 70,
 'max_features': 2,
 'min_samples_leaf': 3,
 'min_samples_split': 8,
 'n_estimators': 1000,
 'oob_score': False,
 'random_state': 42}
```

In [116]:
```python
best_grid = grid_search.best_estimator_
```

In [117]:
```python
best_grid
```

Out[117]:
```
RandomForestClassifier

RandomForestClassifier(criterion='entropy', max_depth=70, max_features=2,
                       min_samples_leaf=3, min_samples_split=8,
                       n_estimators=1000, random_state=42)
```

In [118]:
```python
forest_finetuned = forest.set_params(criterion='entropy', max_depth=70, max_features=2,
                        min_samples_leaf=3, min_samples_split=8,
                        n_estimators=1000)
```

In [119]:
```python
model_final = forest_finetuned.fit(x_train, y_train)
y_pred = model_final.predict(x_test)
print(accuracy_score(y_test,y_pred))
```

```
0.640625
```

In [ ]:

In [121]:
```python
# Cross validation
from sklearn.model_selection import cross_val_score

scores = cross_val_score(forest, x_train, y_train, cv=9)
scores
print(scores)
print(scores.mean())
print("accuracy of training data is: ", scores.mean())
```

```
[0.65034965 0.66197183 0.64084507 0.62676056 0.68309859 0.66901408
 0.70422535 0.69014085 0.77464789]
0.6778948750779737
accuracy of training data is:  0.6778948750779737
```

In [ ]:
```python
# Here we can observe that after fine tuning both training and testing accuracy have reduced.
# The model is no more overfitting.
```

In [74]:
```python
difference = pd.DataFrame(y_test - y_pred)
difference.value_counts()
```

Out[74]:
```
quality
 0         203
 1          66
-1          45
-2           4
 2           2
dtype: int64
```

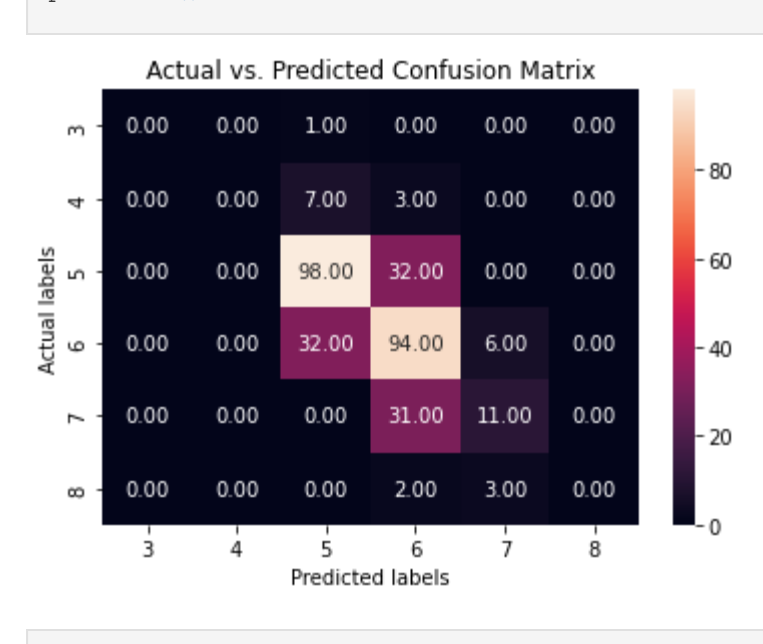
In [75]:
```python
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test,y_pred)
conf_matrix
```

Out[75]:
```
array([[ 0,  0,  1,  0,  0,  0],
       [ 0,  0,  7,  3,  0,  0],
       [ 0,  0, 98, 32,  0,  0],
       [ 0,  0, 32, 94,  6,  0],
       [ 0,  0,  0, 31, 11,  0],
       [ 0,  0,  0,  2,  3,  0]], dtype=int64)
```

In [76]:
```python
ax= plt.subplot()
sns.heatmap(conf_matrix,annot=True, ax= ax, fmt = ".2f")

ax.set_xlabel('Predicted labels');ax.set_ylabel('Actual labels');
ax.set_title('Actual vs. Predicted Confusion Matrix');
ax.xaxis.set_ticklabels([3,4,5,6,7,8]); ax.yaxis.set_ticklabels([3,4,5,6,7,8]);

plt.show()
```



In [ ]:
```python
# From the above confusion matrix we can conclude that the TP for class 5 is 98%, for class 6 is 94%
# and for class 7 it is 11%. For class 5 and 6 the model is predicting with high accuracy.
```

In [122]:
```python
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.00      0.00      0.00        10
           5       0.71      0.76      0.74       130
           6       0.59      0.71      0.65       132
           7       0.55      0.29      0.37        42
           8       0.00      0.00      0.00         5

    accuracy                           0.64       320
   macro avg       0.31      0.29      0.29       320
weighted avg       0.60      0.64      0.61       320
```

In [ ]:
```python
# From the classification report also we can understand that for class 5 and 6 th precision,recall and f1- scor
# as compared to the remaining classes.
```

Another Trial with RandomizedSearchCV

In [34]:
```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [36]:
```python
# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 50)
classifier.fit(x_train, y_train)
```

Out[36]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=50)
```

In [39]:
```python
from sklearn.model_selection import RandomizedSearchCV

from scipy.stats import randint

est = RandomForestClassifier(n_jobs=-1)
rf_p_dist={'max_depth':[3,5,10,None],
              'n_estimators':[10,100,200,300,400,500],
              'max_features':randint(1,12),
               'criterion':['gini','entropy'],
               'bootstrap':[True,False],
               'min_samples_leaf':randint(1,4),
              }
```

In [42]:
```python
def hypertuning_rscv(est, p_distr, nbr_iter,x,y):
    rdmsearch = RandomizedSearchCV(est, param_distributions=p_distr,
                                  n_jobs=-1, n_iter=nbr_iter, cv=9)
    #CV = Cross-Validation ( here using Stratified KFold CV)
    rdmsearch.fit(x,y)
    ht_params = rdmsearch.best_params_
    ht_score = rdmsearch.best_score_
    return ht_params, ht_score

rf_parameters, rf_ht_score = hypertuning_rscv(est, rf_p_dist, 40, x, y)

claasifier=RandomForestClassifier(n_jobs=-1, n_estimators=300,bootstrap= True,criterion='entropy',max_depth=3,m
```

In [44]:
```python
# Predicting the Test set results
y_pred = classifier.predict(x_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)

accuracy_score=accuracy_score(y_test,y_pred)
print(accuracy_score)
from sklearn.model_selection import cross_val_score

cross_val=cross_val_score(claasifier,x,y,cv=10,scoring='accuracy').mean()
print(cross_val)
```

```
0.6525
0.5741155660377358
```

In [ ]:

Conclusion

In [ ]:
```python
# Eventhough tried both GridSearchCV and RandomizedSearchCV, with GridSearchCV better accuracy is getting for b
# training and  testing. RandomizedSearchCV is taking less time for computation as compared with the GridSearch
# Since the results are better with GridSearchCV we can go with that.

# Also the randomforest was overfitting initially. But, doing better after the hyper tuning.

# Final Model Accuracy:

            # For training: 67.789%
            # For testing : 64.065%
```

In [ ]: