dev-bhanushali-assignment1

April 15, 2024

0.1 Imports

[52]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.multiclass import OneVsOneClassifier, OneVsRestClassifier
from sklearn.preprocessing import StandardScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, roc_curve
from sklearn.model_selection import train_test_split
```

1 Original Data

[53]:
```
data = pd.read_csv("./WineQT.csv", index_col="Id")
data.head()
```

[53]:

Id	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68		
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65		
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58		
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56		

```
1       9.8        5
2       9.8        5
3       9.8        6
4       9.4        5
```

1.1 Data preprocessing

check for NA values

[54]:
```
data.isnull().sum()
```

[54]:
```
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
```

1.2 A) Plot histogram of each attribute regarding Y=0, Y=1 and Y=2, and display the number of samples (Y) for each quality classes.

What can you say regarding the quality (Y) classes distribution? What is your conclusion regarding the expected performance of the classifier?

The quality classes have been relabeled from numeric to categorical type using the following label encoder mapping

Quality (Integer)	Label (Categorical)
2	0
3	0
4	0
5	1
6	1
7	2
8	2

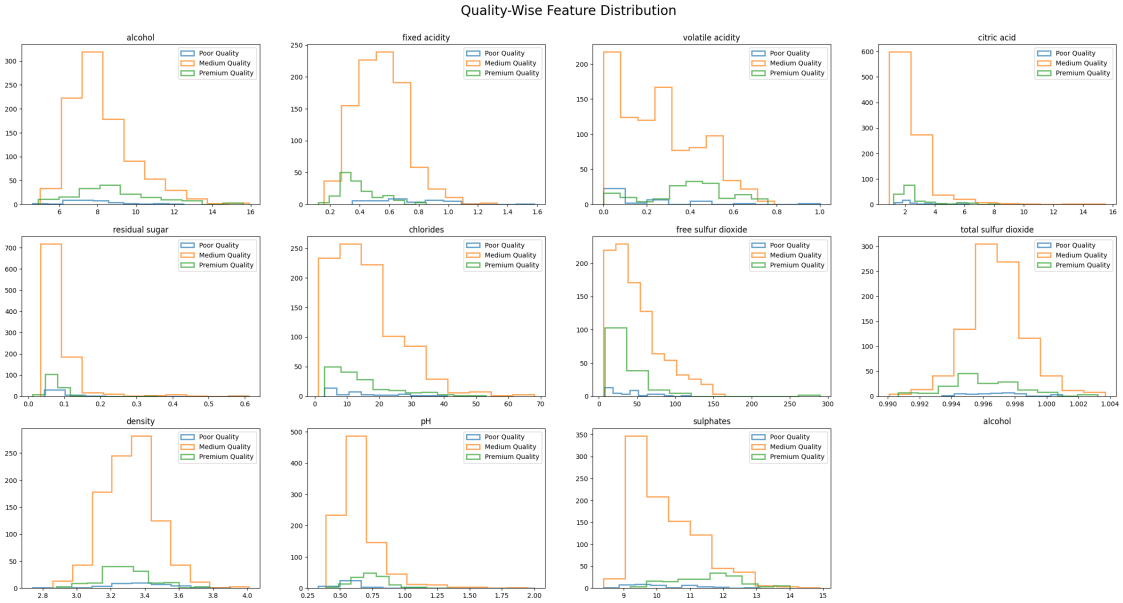
segregation of dataframe rows as per the categorical classes of the quality feature

[55]:
```
class_0 = data[(data["quality"] == 2) | (data["quality"] == 3) |␣
↪(data["quality"] == 4)]
```

```
class_1 = data[(data["quality"] == 5) | (data["quality"] == 6)]
class_2 = data[(data["quality"] == 7) | (data["quality"] == 8)]

quality_data = {"Poor Quality": class_0, "Medium Quality": class_1, "Premium␣
 ↪Quality": class_2}
```

Quality-Wise feature distribution

[56]:
```
plt.figure(figsize=(30, 15))
plt.title("Quality-Wise Feature Distribution", pad=45, fontdict={"size":20})
for quality in quality_data.keys():
    cols = quality_data[quality].columns[:-1]
    class_data = quality_data[quality]

    plt.ylabel("Frequency")
    plt.axis("off")

    for i in range(4):
        for j in range(3):
            idx = (i*3) + j
            col_name = cols[idx - 1]
            if(idx + 1 == 12):
                plt.subplot(3,4,idx + 1), plt.hist(class_data[cols[idx - 1]],␣
 ↪density=True, alpha=0), plt.title(col_name)
            else:
                plt.subplot(3,4,idx + 1), plt.hist(class_data[cols[idx]],␣
 ↪alpha=0.7, histtype="step", linewidth=2, label=str(quality)), plt.
 ↪title(col_name)
                plt.legend(["Poor Quality", "Medium Quality", "Premium␣
 ↪Quality"])
```

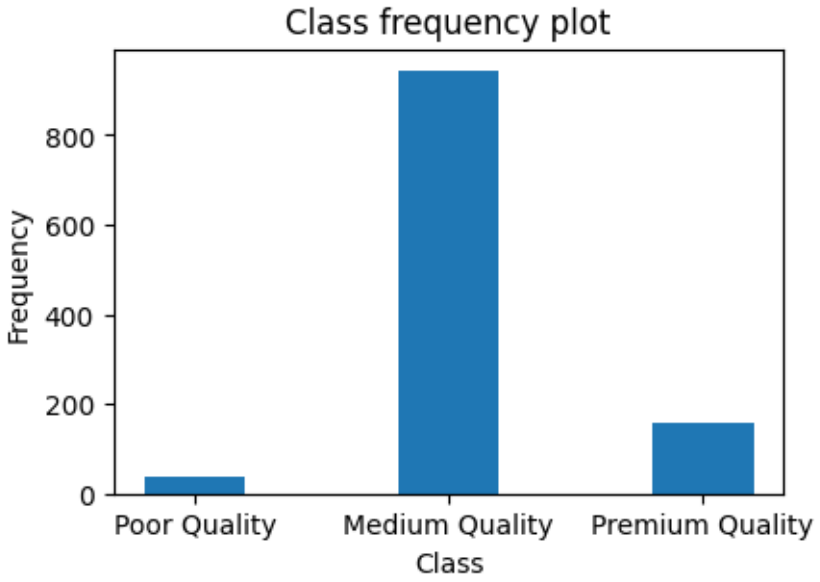


Class frequency plot

[57]:
```
class_sample_distribution = {}
for key in quality_data.keys():
    class_sample_distribution[key] = len(quality_data[key])
```

[58]:
```
plt.figure(figsize=(4.5,3))
plt.title("Class frequency plot")
plt.xlabel("Class"), plt.ylabel("Frequency")
plt.bar(class_sample_distribution.keys(), class_sample_distribution.values(),␣
↪width=0.4)
```

[58]: <BarContainer object of 3 artists>



Answering Questions What can you say regarding the quality (Y) classes distribution?

It is clear that the classes are unequally distributed. There is a large number of medium quality wine samples and a very low number of samples of the poor and premium quality wine.

What is your conclusion regarding the expected performance of the classifier?

Due to the biased nature of class sample frequencies, it is expected that the model will generalize poorly on real life data. The model may have high accuracy, in this case by classifiying the majority class correctly but has low classification accuracy for the poor and premium class when seen indivisually.

1.3 B) Perform one run of modeling and test. Compare the obtained test accuracy by using:

1. One versus All Classifier And
2. One versus One Classifier

Label Encoding The quality classes have been relabeled from numeric to categorical type using the following label encoder mapping

Quality (Integer)	Label (Categorical)
2	0
3	0
4	0
5	1
6	1
7	2
8	2

[59]:
```
map = {2: 0, 3: 0, 4: 0, 5:1, 6:1, 7:2, 8:2}
quality = data["quality"]
encoded_quality = quality.apply(lambda x: map[x])
data["Encoded_Quality"] = encoded_quality

data.head()
```

[59]:

Id	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Encoded_Quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	1
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	1
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58			
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56			

```
3       9.8         6                 1
4       9.4         5                 1
```

generate dataset for train and test using stratified sampling to keep same ratios of target classes in train and test

[60]:
```
X, y = data[data.columns[:-2]], data[data.columns[-1]]
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪stratify=y, random_state=3)
```

feature scaling

[61]:
```
scaler=StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

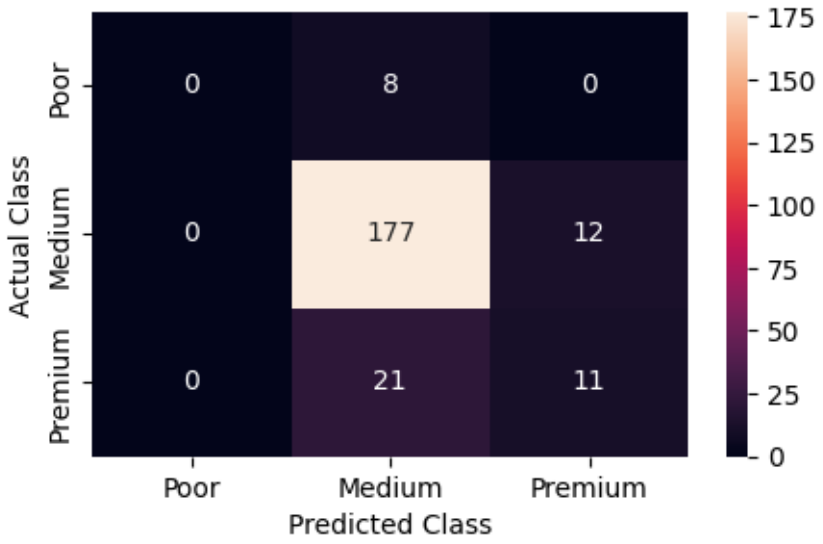
One VS One Classification

[62]:
```
ovo = OneVsOneClassifier(
    LogisticRegression(max_iter=500, penalty='l2')
).fit(x_train, y_train)

y_pred = ovo.predict(x_test)
```

confusion matrix

[63]:
```
cf_martix_df = pd.DataFrame(confusion_matrix(y_pred=y_pred, y_true=y_test),␣
 ↪index = ["Poor", "Medium", "Premium"],
                  columns = ["Poor", "Medium", "Premium"])
plt.figure(figsize = (5,3))
heatmap = sns.heatmap(cf_martix_df, annot=True, fmt="g")
heatmap.set_xlabel("Predicted Class")
heatmap.set_ylabel("Actual Class")
plt.plot()
```

[63]:
```
[]
```



here we can see that all predictions for the poor class were incorrect and more than half the predictions of premium class were also incorrect which is not desirable, even though we have a high accuracy (below) supported majorly by the medium quality class due to heavy class imbalance (Model can classify the medium quality class correctly often due to the high number of samples)

classification report

[64]:
```
print(classification_report(y_pred=y_pred, y_true=y_test))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.86      0.94      0.90       189
           2       0.48      0.34      0.40        32

    accuracy                           0.82       229
   macro avg       0.45      0.43      0.43       229
weighted avg       0.78      0.82      0.80       229


c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

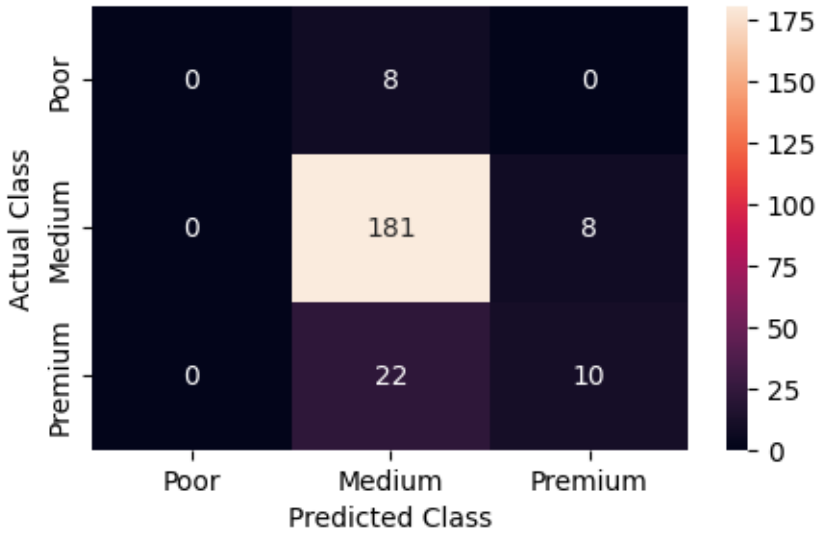
One VS Rest Classification

[65]:
```
ovo = OneVsRestClassifier(
    LogisticRegression(penalty='l2')
).fit(x_train, y_train)

y_pred = ovo.predict(x_test)
```

confusion matrix

[66]:
```
cf_martix_df = pd.DataFrame(confusion_matrix(y_pred=y_pred, y_true=y_test),␣
 ↪index = ["Poor", "Medium", "Premium"],
                  columns = ["Poor", "Medium", "Premium"])
plt.figure(figsize = (5,3))
heatmap = sns.heatmap(cf_martix_df, annot=True, fmt="g")
heatmap.set_xlabel("Predicted Class")
heatmap.set_ylabel("Actual Class")
plt.plot()
```

[66]:
```
[]
```



here we can see that all predictions for the poor class were incorrect and more than half the predictions of premium class were also incorrect which is not desirable, even though we have a high accuracy supported majorly by the medium class due to heavy class imbalance

classification report

[67]:
```
print(classification_report(y_pred=y_pred, y_true=y_test))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.86      0.96      0.90       189
           2       0.56      0.31      0.40        32

    accuracy                           0.83       229
   macro avg       0.47      0.42      0.43       229
weighted avg       0.79      0.83      0.80       229
```

```
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

2 New Data

[68]:
```
data_new = pd.read_csv("./Wine_Test_02_6_8_red.csv")
data_new.head()
```

[68]:
```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            8.5              0.28         0.56             1.8      0.092
1            7.9              0.32         0.51             1.8      0.341
2            7.6              0.39         0.31             2.3      0.082
3            8.1              0.38         0.28             2.1      0.066
4            7.3              0.45         0.36             5.9      0.074

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 35.0                 103.0   0.9969  3.30       0.75
1                 17.0                  56.0   0.9969  3.04       1.08
2                 23.0                  71.0   0.9982  3.52       0.65
3                 13.0                  30.0   0.9968  3.23       0.73
```

```
4                12.0                 87.0   0.9978  3.33       0.83
```

```
   alcohol  quality
0     10.5        2
1      9.2        1
2      9.7        1
3      9.7        2
4     10.5        1
```

2.1 Data preprocessing

check for NULL values

[69]:
```
data_new.isnull().sum()
```

[69]:
```
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
```

2.2 Data Visualization

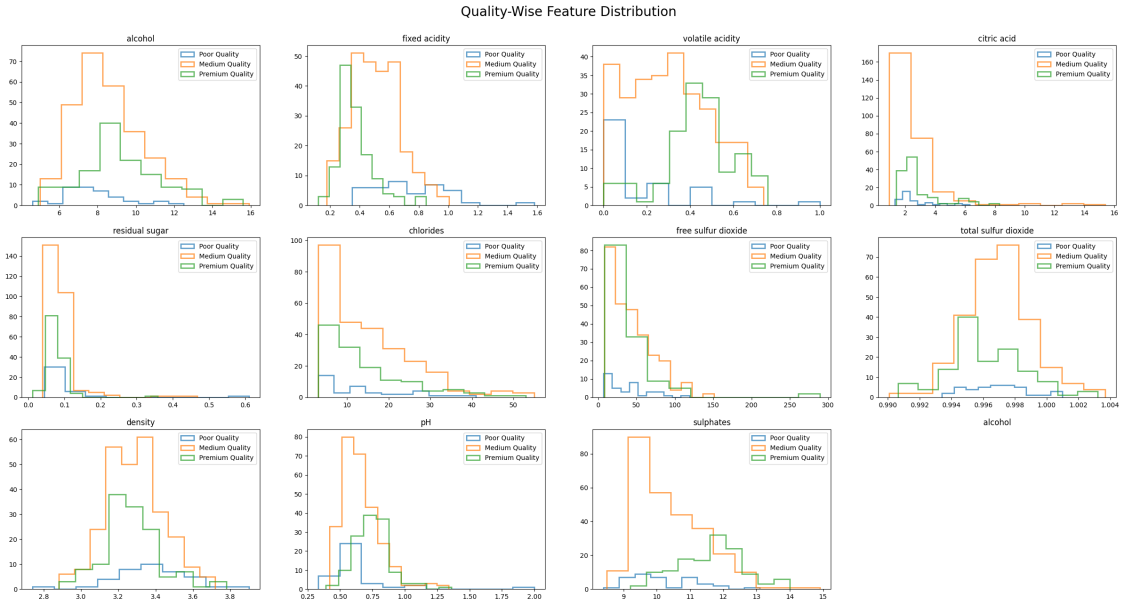
segregation of dataframe rows as per the categorical classes of the quality feature

[70]:
```
class_0 = data_new[(data_new["quality"] == 0)]
class_1 = data_new[(data_new["quality"] == 1)]
class_2 = data_new[(data_new["quality"] == 2)]

quality_data = {"Poor Quality": class_0, "Medium Quality": class_1, "Premium␣
 ↪Quality": class_2}
```

Quality-Wise feature distribution

[71]:
```
plt.figure(figsize=(30, 15))
plt.title("Quality-Wise Feature Distribution", pad=45, fontdict={"size":20})
for quality in quality_data.keys():
    cols = quality_data[quality].columns[:-1]
    class_data = quality_data[quality]
```

```
    plt.ylabel("Frequency")
    plt.axis("off")

    for i in range(4):
        for j in range(3):
            idx = (i*3) + j
            col_name = cols[idx - 1]
            if(idx + 1 == 12):
                plt.subplot(3,4,idx + 1), plt.hist(class_data[cols[idx - 1]],␣
↪density=True, alpha=0), plt.title(col_name)
            else:
                plt.subplot(3,4,idx + 1), plt.hist(class_data[cols[idx]],␣
↪alpha=0.7, histtype="step", linewidth=2, label=str(quality)), plt.
↪title(col_name)
                plt.legend(["Poor Quality", "Medium Quality", "Premium␣
↪Quality"])
```



Class frequency plot

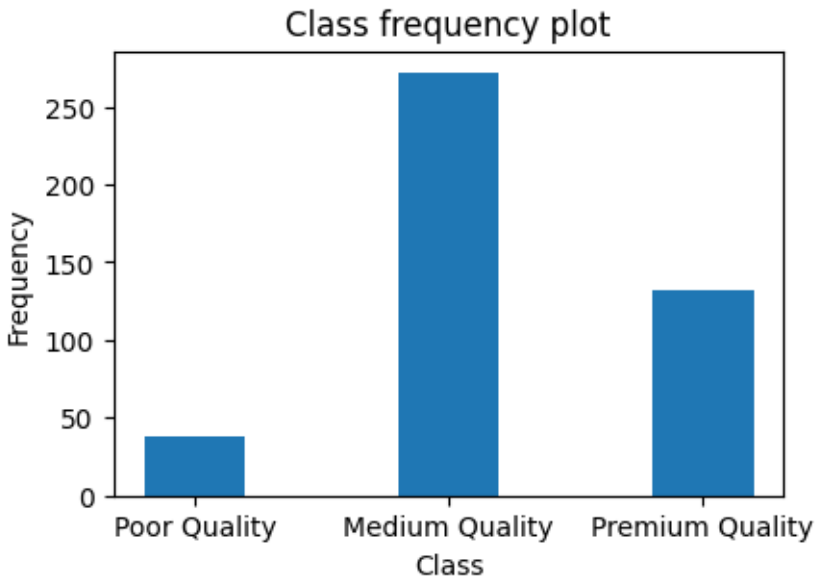
[72]:
```
class_sample_distribution = {}
for key in quality_data.keys():
    class_sample_distribution[key] = len(quality_data[key])
```

[73]:
```
class_sample_distribution
```

[73]:
```
{'Poor Quality': 38, 'Medium Quality': 272, 'Premium Quality': 132}
```

[74]:
```
plt.figure(figsize=(4.5,3))
plt.title("Class frequency plot")
plt.xlabel("Class"), plt.ylabel("Frequency")
plt.bar(class_sample_distribution.keys(), class_sample_distribution.values(),␣
↪width=0.4)
```

[74]:
```
<BarContainer object of 3 artists>
```



Answering Questions What can you say regarding the quality (Y) classes distribution?

the class distribution is still imbalanced but is somewhat better than the older dataset since there are relavtively more samples of each class and the ratios of sample count difference for each class arent as high as the previous dataset

What is your conclusion regarding the expected performance of the classifier?

Due to the biased nature of class samples, it is expected that the model will generalize poorly on real life data. The model may have high accuracy, in this case by classifiying the majority class correctly but will end up having low classification accuracy for the poor and medium class when seen indivisually.

It is probable that this dataset will lead to better results compared to the previous dataset due to better sample ratios but because the distribution ratios are still not very good, the disadvantages will out weigh the improvements and will lead to bad performance

2.3 B) Perform one run of modeling and test. Compare the obtained test accuracy by using:

1. One versus All Classifier And
2. One versus One Classifier

generate dataset for train and test using stratified sampling to keep same ratios of classes in train and test

[75]:
```
X_new, y_new = data_new[data_new.columns[:-1]], data_new[data_new.columns[-1]]
x_train, x_test, y_train, y_test = train_test_split(X_new, y_new, test_size=0.
↪2, stratify=y_new, random_state=3)
```

[76]:
```
(data_new[data_new.columns[-1]] == y_new).value_counts()
y_new.value_counts()
```

[76]:
```
quality
1    272
2    132
0     38
Name: count, dtype: int64
```

feature scaling

[77]:
```
scaler=StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

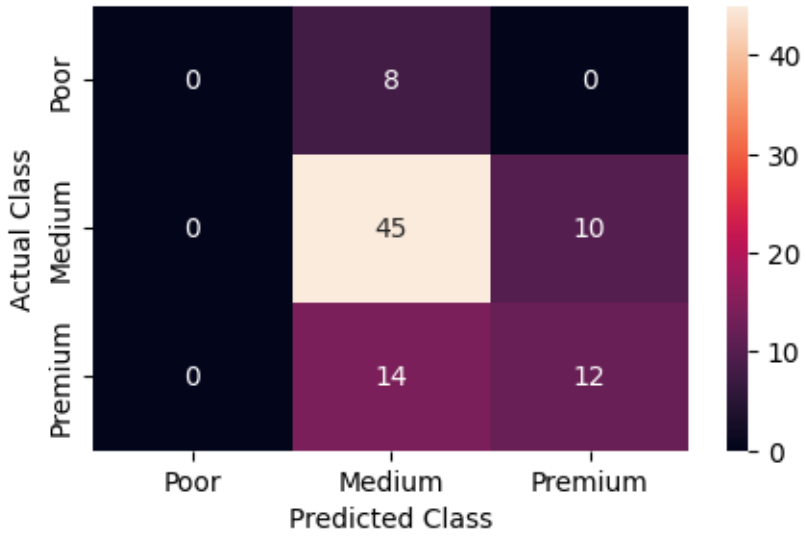
One VS One Classification

[78]:
```
ovo = OneVsOneClassifier(
    LogisticRegression(max_iter=500, penalty='l2')
).fit(x_train, y_train)

y_pred = ovo.predict(x_test)
```

confusion matrix

[79]:
```
cf_martix_df = pd.DataFrame(confusion_matrix(y_pred=y_pred, y_true=y_test),␣
↪index = ["Poor", "Medium", "Premium"],
                  columns = ["Poor", "Medium", "Premium"])
plt.figure(figsize = (5,3))
heatmap = sns.heatmap(cf_martix_df, annot=True, fmt="g")
heatmap.set_xlabel("Predicted Class")
heatmap.set_ylabel("Actual Class")
plt.plot()
```

[79]:
```
[]
```



here we can see that all predictions for the poor class were incorrect and more than half the predictions of premium class were also incorrect which is not desirable.

classification report

[80]:
```
print(classification_report(y_pred=y_pred, y_true=y_test))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.67      0.82      0.74        55
           2       0.55      0.46      0.50        26

    accuracy                           0.64        89
   macro avg       0.41      0.43      0.41        89
weighted avg       0.57      0.64      0.60        89


c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
```

```
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

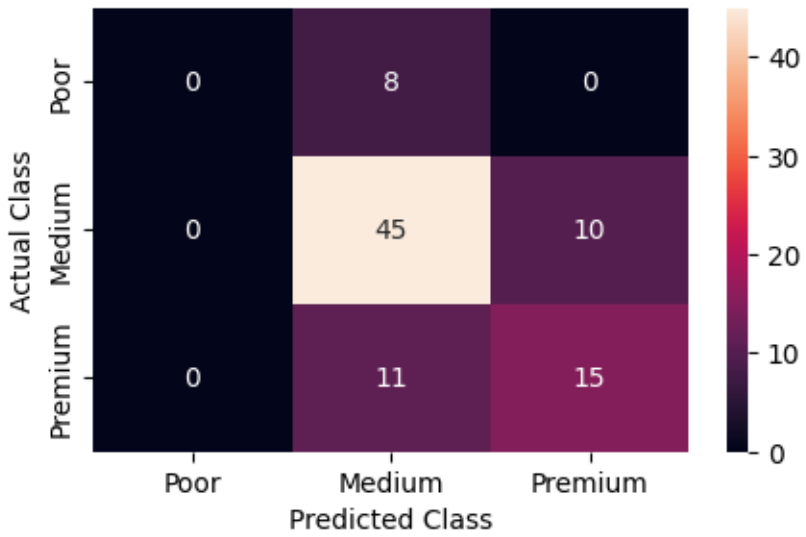
One VS Rest Classification

[81]:
```
ovo = OneVsRestClassifier(
    LogisticRegression(penalty='l2')
).fit(x_train, y_train)

y_pred = ovo.predict(x_test)
```

confusion matrix

[82]:
```
cf_martix_df = pd.DataFrame(confusion_matrix(y_pred=y_pred, y_true=y_test),␣
 ↪index = ["Poor", "Medium", "Premium"],
                  columns = ["Poor", "Medium", "Premium"])
plt.figure(figsize = (5,3))
heatmap = sns.heatmap(cf_martix_df, annot=True, fmt="g")
heatmap.set_xlabel("Predicted Class")
heatmap.set_ylabel("Actual Class")
plt.plot()
```

[82]:
```
[]
```



here we can see that all predictions for the poor class were incorrect and more close to half the predictions of premium class were also incorrect which is not desirable.

classification report

[83]:
```
print(classification_report(y_pred=y_pred, y_true=y_test))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.70      0.82      0.76        55
           2       0.60      0.58      0.59        26

    accuracy                           0.67        89
   macro avg       0.43      0.47      0.45        89
weighted avg       0.61      0.67      0.64        89

c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
c:\Users\devbh\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

3 Observations

It can be seen that the accuracy decreases in the second dataset which is reasonable due to the lower number of samples across classes, but the model also learns better in the second dataset since the class-wise ratio of class samples are not that huge and there are lesser true negatives for the predictions for class 3.

It can be concluded that with better class distribution, it is likely that the performance will increase in contrast to when the class distributions are highly biased