Name: Peter Bozzo

The micro and macro technique for calculating the precision, recall and F score was initially used to evaluate this model.
The following was the formula used for calculating micro precision, recall and F Score:

$$P = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FP_c}$$

$$R = \frac{\sum_c TP_c}{\sum_c TP_c + \sum_c FN_c}$$

Macro precision, recall and F score:

$$Precision_M = \frac{\sum_{i=1}^{c} Precision(i)}{c} \qquad Recall_M = \frac{\sum_{i=1}^{c} Recall\ (i)}{c}$$

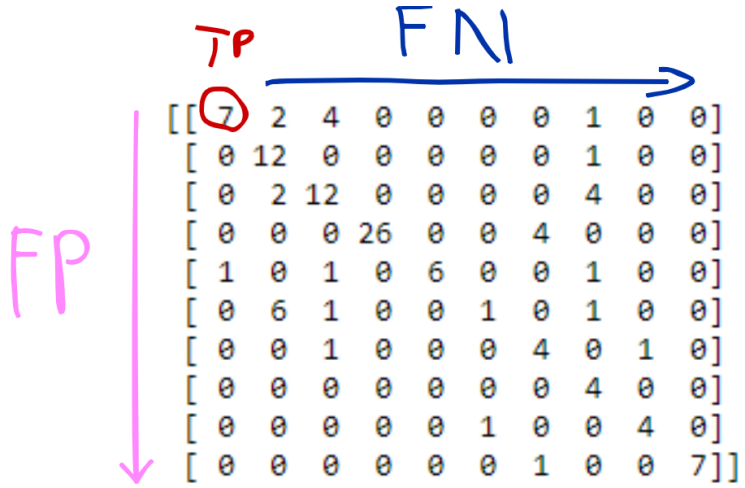
From our code, it was observed that micro averaging gives the same result for all 3 attributes.
While macro averaging shows similar scores off roughly 0.7 for precision, recall and F score.

```
micro precision: 0.7155172413793104  micro recall: 0.7155172413793104  micro F1: 0.7155172413793104

macro precision: 0.7129811270600743  macro recall: 0.7075854700854701  macro F1: 0.710273051564907

weighted_average_precision: 0.7162027489046546  weighted_average_recall: 0.6545184939581492
```

Upon further evaluating and research, it was realised that for a multi-class classifier, when calculating True-Negatives (TN), True-Positives (TP), False-Positive (FP), False-Negative (FN) from a Confusion Matrix, the resulting FN = FP.

```
[[ 7  2  4  0  0  0  0  1  0  0]
 [ 0 12  0  0  0  0  0  1  0  0]
 [ 0  2 12  0  0  0  0  4  0  0]
 [ 0  0  0 26  0  0  4  0  0  0]
 [ 1  0  1  0  6  0  0  1  0  0]
 [ 0  6  1  0  0  1  0  1  0  0]
 [ 0  0  1  0  0  0  4  0  1  0]
 [ 0  0  0  0  0  0  0  4  0  0]
 [ 0  0  0  0  0  1  0  0  4  0]
 [ 0  0  0  0  0  0  1  0  0  7]]
```

FN is equal to FP because as you traverse the matrix to aggregate scores for TP, TN, FP, FN, every False instance is FP for a class, and at the same time every Negative instance will be a FN for a class.



This results in both precision and recall to always be equal based on their respective formulas. Since precision and recall are now equivalent, F score will now also be equivalent since F score is simply the sum of average of all precisions and recalls. Another way of showing this is by transformation and reduction of the F score formula:

$$F1 = 2\frac{P * R}{P + R} = 2\frac{P * P}{P + P} = 2\frac{P^2}{2P} = \frac{P^2}{P} = P$$

F1 = P, hence F1 = P = R.

Following our result of micro averaging, a decision was made to perform a weighted average of precision and recall on the model since micro averaging did not provide any valuable information. Upon completion, it was found that both weighted average precision and recall scores were congregated at around 0.7 and 0.65 respectively.
Macro average recall does not take into account weighted bias in the dataset, meaning that a class that has 400 instances is weighted the same as a class with 50 instances.
Hence even with high values for both macro precision and recall, it is not completely deterministic of the model performance. However, if one was to observe the Confusion Table, we can see that the majority of predicted classes are in the TP diagonal lines, with very few outer errors. This in the least was a great indication that our Naive Bayes model performed well overall. The original accuracy of the model was:

```
accuracy:  71.55172413793103
```

Euclidean distance between each key pair was the chosen characteristic for each new feature. This was done by employing the sklearn euclidean_distance() function that calculates the distance of each pair of points to each other pair of points within the given dataset. The resulting 2D array stores 11x11 columns, since we are comparing each distance to all of the others. The 2D array has a dimension of 121x747, a significant increase in size from the original 22x747. Due to the large increase in the number of attributes within our dataset, Chi Squared was chosen as the feature selection technique, in order to select the most correlated features to the given class labels from our dataset. K = 22 was chosen for Chi Squared, this was to keep the numbers of attributes selected to be equal to the original dataset. After Chi Squared evaluation, the resulting 2D array shows:

	feature1_x2	feature2_x2	feature3_x2	feature4_x2	feature5_x2	\
0	121.456106	123.961773	132.815286	86.724648	112.039619	
1	9.399098	39.152864	90.070921	95.283228	205.292640	
2	112.668433	96.623463	131.224467	204.761463	126.984569	
3	122.302388	210.699560	218.018303	217.027645	168.004357	
4	192.319403	191.993091	215.233186	188.849239	117.033427	
..	...	...	...	...	...	
742	48.448869	189.132938	285.619807	202.327890	258.284194	
743	93.588034	209.392603	246.115041	305.197945	280.622121	
744	113.072668	224.422076	244.408034	340.918365	272.759912	
745	27.009445	152.421351	216.475837	216.675874	251.382615	
746	72.454321	186.378063	220.949403	276.125933	279.950657	

	feature6_x2	feature7_x2	feature8_x2	feature9_x2	feature10_x2	...	\
0	99.423570	139.247485	78.918661	118.160739	99.423570	...	
1	57.126067	27.109553	12.756109	30.132535	57.126067	...	
2	108.184771	101.232863	10.993585	15.568378	108.184771	...	
3	54.157991	60.757601	80.080250	87.936493	54.157991	...	
4	207.252991	69.319602	126.020347	61.055029	207.252991	...	
..	...	...	...	...	...	...	
742	267.787800	134.497900	51.525298	82.708434	267.787800	...	
743	301.843344	145.417426	58.801076	99.156259	301.843344	...	
744	293.797496	161.016085	55.334086	78.035250	293.797496	...	
745	241.035405	120.304799	49.093992	71.842309	241.035405	...	
746	298.831444	151.413257	57.981461	91.294917	298.831444	...	

	feature13_x2	feature14_x2	feature15_x2	feature16_x2	feature17_x2	\
0	139.247485	118.160739	35.550573	123.961773	132.815286	
1	27.109553	30.132535	155.258813	39.152864	90.070921	
2	101.232863	15.568378	108.717640	96.623463	131.224467	
3	60.757601	87.936493	113.606739	210.699560	218.018303	
4	69.319602	61.055029	27.830552	191.993091	215.233186	
..	...	...	...	...	...	
742	134.497900	82.708434	233.571900	189.132938	285.619807	
743	145.417426	99.156259	234.016892	209.392603	246.115041	
744	161.016085	78.035250	232.684831	224.422076	244.408034	
745	120.304799	71.842309	225.693110	152.421351	216.475837	
746	151.413257	91.294917	238.722788	186.378063	220.949403	

	feature18_x2	feature19_x2	feature20_x2	feature21_x2	feature22_x2
0	46.294615	112.039619	35.550573	86.724648	46.294615
1	76.688826	205.292640	155.258813	95.283228	76.688826
2	107.480887	126.984569	108.717640	204.761463	107.480887
3	10.553161	168.004357	113.606739	217.027645	10.553161
4	27.692464	117.033427	27.830552	188.849239	27.692464
..	...	...	...	...	...
742	296.872412	258.284194	233.571900	202.327890	296.872412
743	292.374398	280.622121	234.016892	305.197945	292.374398
744	300.508771	272.759912	232.684831	340.918365	300.508771
745	210.544562	251.382615	225.693110	216.675874	210.544562
746	245.598993	279.950657	238.722788	276.125933	245.598993

[747 rows x 22 columns]

The engineered features were passed into the Naiive Bayes algorithm, during the testing face, the test() function had to also perform feature engineering on the test inputs to ensure consistency during testing. The accuracy that the system gave is:

```
accuracy:   29.310344827586203
```

This was significantly lower than the original accuracy obtained without feature engineering. Euclidean distance between each pair of points may not have been a better predictor than the x and y asyms of each key point, hence making the model more error prone.