Model evaluation recap

Classification

Confusion matrix

axis 1

axis 0

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

Predicted Class

Actual Class

		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Confusion matrix

```
[16]: predictions = logmodel.predict(X_test)
```

** Create a classification report for the model.**

```
[17]: from sklearn.metrics import classification_report
```

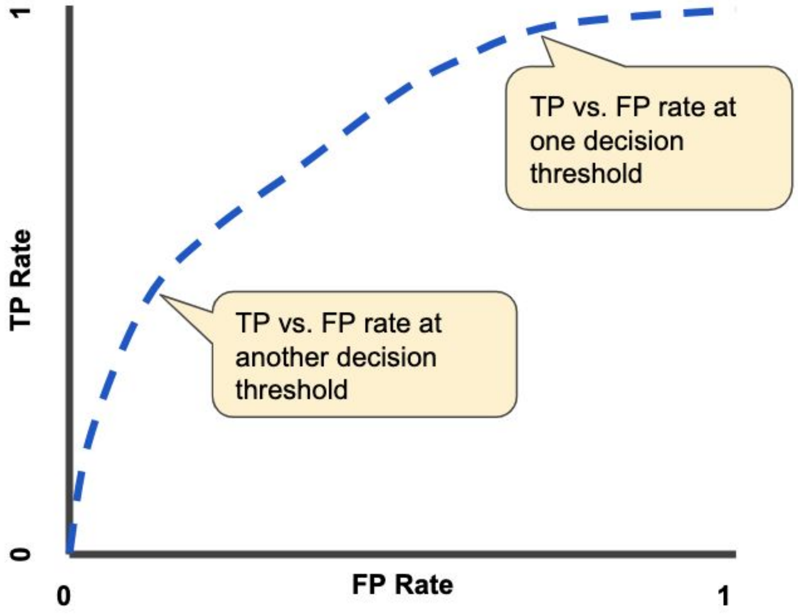
```
[18]: print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.87      0.96      0.91       162
           1       0.96      0.86      0.91       168

    accuracy                           0.91       330
   macro avg       0.91      0.91      0.91       330
weighted avg       0.91      0.91      0.91       330
```

```
[23]: from sklearn.metrics import confusion_matrix
      confusion_matrix(y_test,predictions)
```

```
[23]: array([[156,   6],
             [ 24, 144]])
```

ROC curve and AUC area; the biggest the AUC the better



True Positive Rate (**TPR**) is a synonym for recall

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR)

$$FPR = \frac{FP}{FP + TN}$$

Regressions

Should I use MAE or MSE/RMSE?

$$MAE = \frac{1}{samples} \sum_{i=0}^{samples-1} |y_i - \hat{y}_i|$$

$$MSE = \frac{1}{samples} \sum_{i=0}^{samples-1} (y_i - \hat{y}_i)^2$$

It depends on the nature of the decisions derived.

Recommendation; If you don't have strong preference clean your dataset and use MSE.

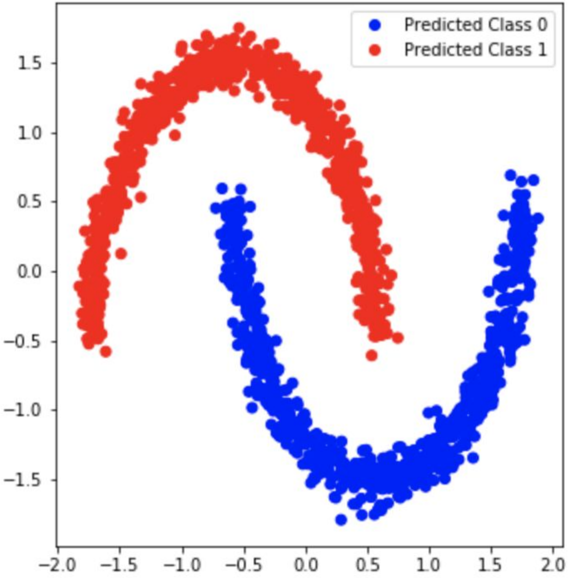
Unsupervised evaluation

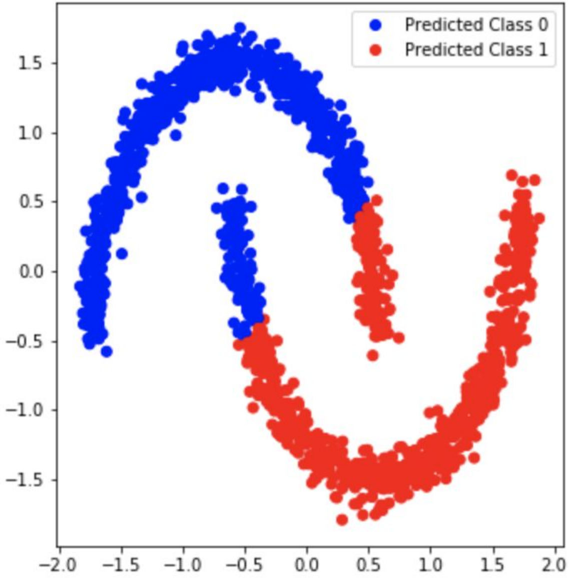
Silhouette score

```
from sklearn.metrics import silhouette_score
score = silhouette_score(X, labels, metric='euclidean')
```

Euclidean distance (the straight line / shortest distance between two points) is the simplest, but it **does not work well for high dimensional data**.

The score is in the range [-1, 1], with a higher score corresponding to dense, well-separated clusters and scores around 0 indicating overlapping clusters



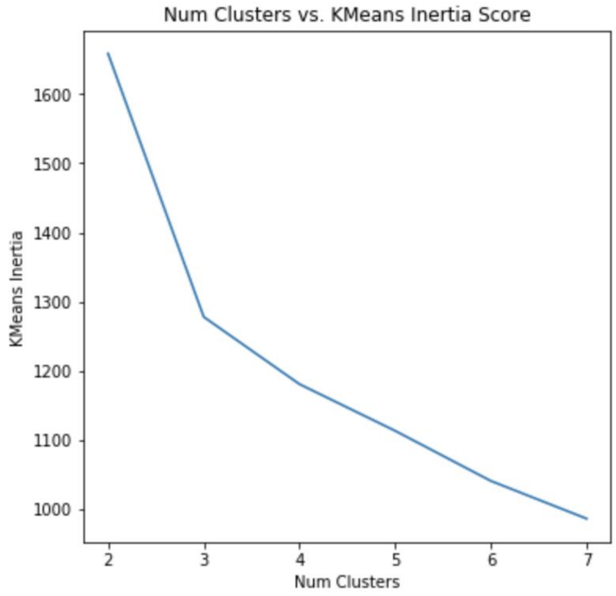



```
Model 1 Silhouette Score: 0.39502725867409694
Model 2 Silhouette Score: 0.49730694704725587
```

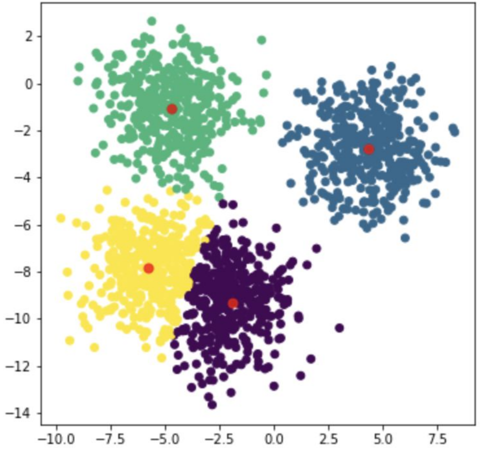
Optimal amount of clusters to be set as hyperparameter

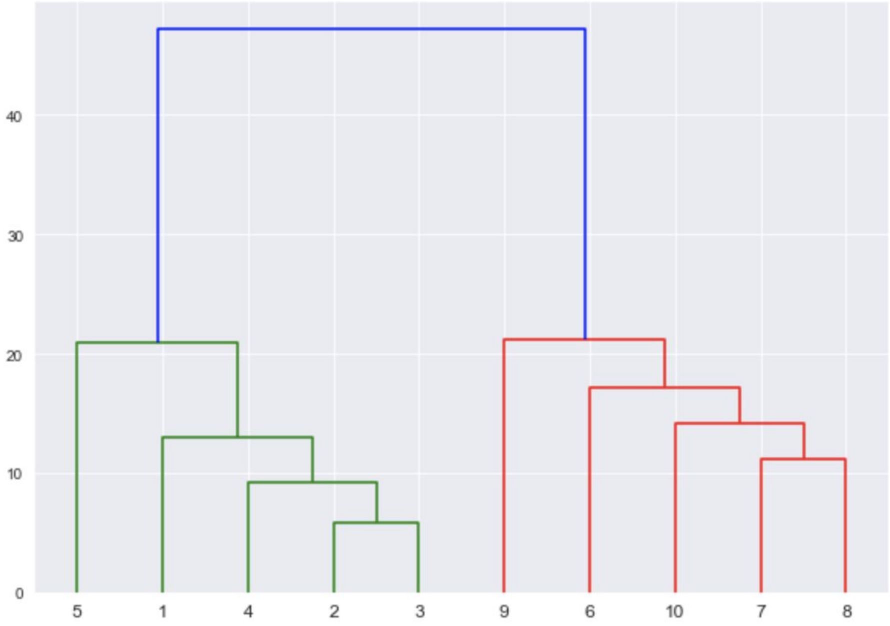
```
from yellowbrick.cluster import KElbowVisualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(4,12))
visualizer.fit(X)
visualizer.poof()
```

The optimal number of clusters is where the plot displays an "elbow" or inflection point.



Look at it visually to assess on proper amount of clusters





Let's see some code :D