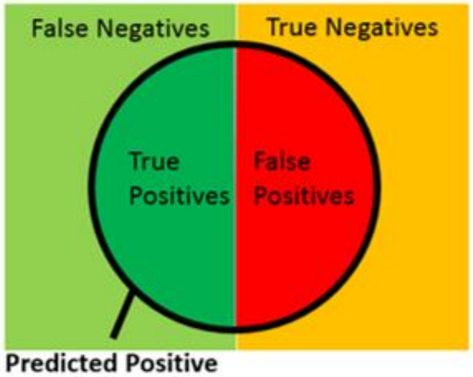
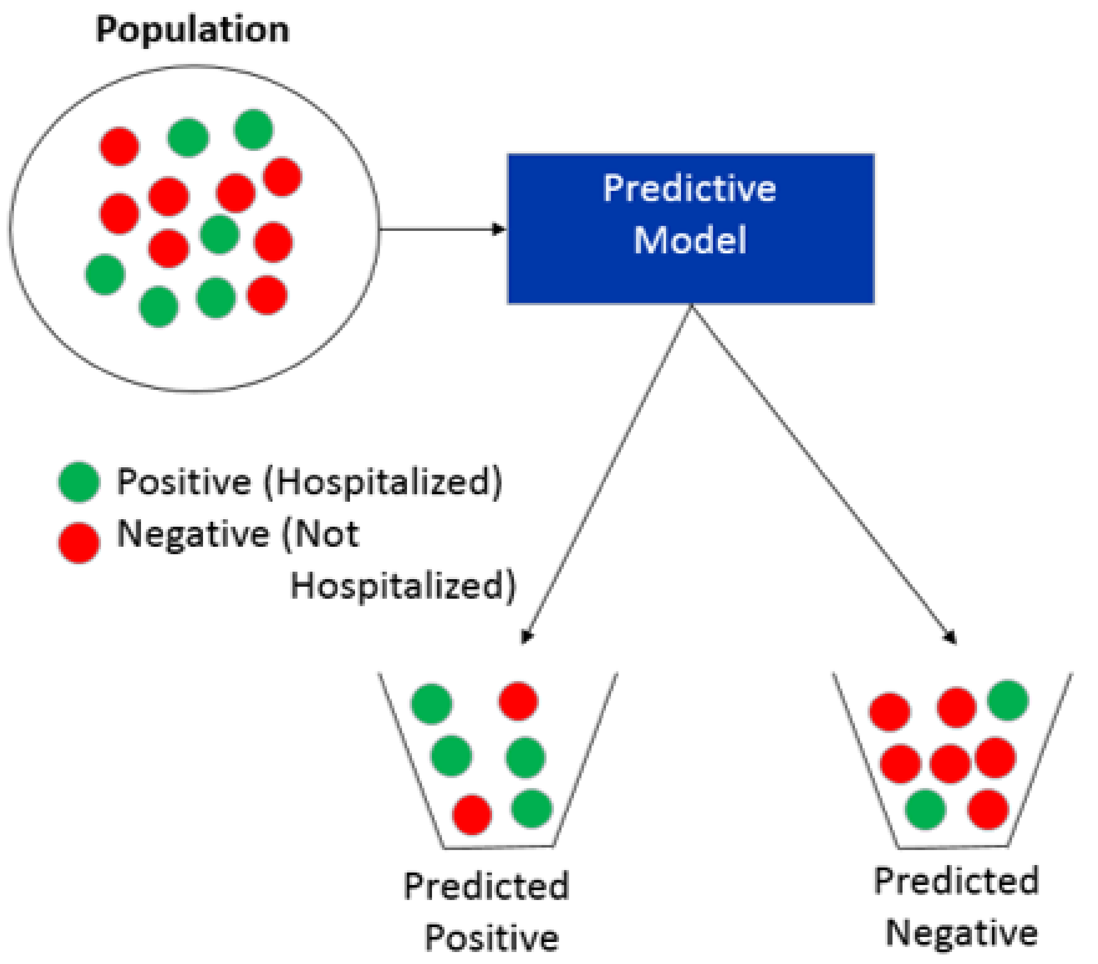
Classification Metrics in Scikit Learn

Metrics in Predictive Modelling

- One major area of predictive modeling in data science is classification. Classification consists of trying to predict which class a particular sample from a population comes from.
- For example, if we are trying to predict if a particular patient will be re-hospitalized, the two possible classes are hospital (positive) and not-hospitalized (negative).
- The classification model then tries to predict if each patient will be hospitalized or not hospitalized.
- In other words, classification is simply trying to predict which bucket (predicted positive vs predicted negative) a particular sample from the population should be placed as seen below.



Metrics in Predictive Modelling



Metrics in Predictive Modelling

- **True Positives:** people that are hospitalized that you predict will be hospitalized
- **True Negatives:** people that are NOT hospitalized that you predict will NOT be hospitalized
- **False Positives:** people that are NOT hospitalized that you predict will be hospitalized
- **False Negatives:** people that are hospitalized that you predict will NOT be hospitalized

Metrics for Evaluating Performance of the Models

1. Accuracy Score Metric

Accuracy_score which is imported as

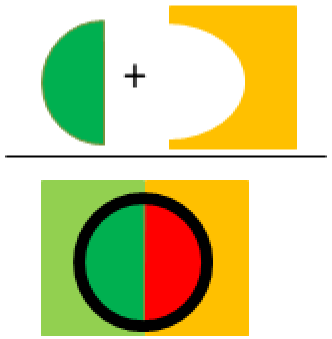
- from sklearn.metrics import accuracy_score

returns “accuracy classification score”. What it does is the calculation of “How accurate the classification is”

- It is the most common metric for classification which is the fraction of samples predicted correctly as shown below:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} =$$

Fraction predicted correctly



Metrics for Evaluating Performance of the Models

- Presented as a percentage by multiplying the result by 100.

  classification accuracy = correct predictions / total predictions * 100

- Classification accuracy can also easily be turned into a misclassification rate or error rate by inverting the value, such as:

  error rate = (1 - (correct predictions / total predictions)) * 100

Metrics for Evaluating Performance of the Models

We can obtain the accuracy score from scikit-learn, which takes as inputs the actual labels and the predicted labels

- from sklearn.metrics import accuracy_score
- accuracy_score(df.actual_label.values, df.predicted_RF.values)

- Shows answer like 0.6705165630156111

Classification Accuracy limitations

- Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
# Make predictions on validation dataset
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
print(predictions)
print("Accuracy Score :", accuracy_score(Y_validation, predictions))
```

```
['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-
versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor'
'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-
virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-
virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor' 'Iris-
virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-
virginica' 'Iris-virginica']
 Accuracy Score : 0.9
```

concat No results Options

jupyter Diabetes Analysis & Prediction Using ML Last Checkpoint: 02/19/2019 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
56      1
Name: Outcome, Length: 192, dtype: int64
```

Linear SVM

In [42]:

```
model=svm.SVC(kernel='linear')
model.fit(train_X,train_Y)
prediction=model.predict(test_X)
print('Accuracy of linear svm =',metrics.accuracy_score(prediction,test_Y))
```

```
Accuracy of linear svm = 0.7708333333333334
```

RBF SVM

In [43]:

```
from sklearn.metrics import accuracy_score
model=svm.SVC(kernel='rbf')

model.fit(train_X,train_Y)
prediction=model.predict(test_X)
print('Accuracy of rbf svm =',accuracy_score(prediction,test_Y))
```

```
Accuracy of rbf svm = 0.6510416666666666
```

Metrics for Evaluating Performance of the Models

2. Confusion Matrix

- A clean and unambiguous way to present the summary of prediction results of a classifier.
- Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.
- The number of correct and incorrect predictions are summarized with count values and broken down by each class
- **The confusion matrix shows the ways in which your classification model is confused when it makes predictions.**

Metrics for Evaluating Performance of the Models

Process for calculating a confusion Matrix

- You need a test dataset or a validation dataset with expected outcome values.
- Make a prediction for each row in your test dataset.
- From the expected outcomes and predictions count:
  - The number of correct predictions for each class.
  - The number of incorrect predictions for each class, organized by the class that was predicted.

Metrics for Evaluating Performance of the Models

- These numbers are then organized into a table, or a matrix as follows:
- **Expected down the side**: Each row of the matrix corresponds to a predicted class.
- **Predicted across the top**: Each column of the matrix corresponds to an actual class.
- The counts of correct and incorrect classification are then filled into the table.

Confusion Matrix		Actual	
		Hospitalized	Not Hospitalized
Predicted	Hospitalized	33	10
	Not Hospitalized	17	40

2-Class Confusion Matrix Case Study

- Let’s pretend we have a two-class classification problem of predicting whether a photograph contains a man or a woman.
- We have a test dataset of 10 records with expected outcomes and a set of predictions from our classification algorithm.

Expected, Predicted
man, woman
man, man
woman, woman
man, man
woman, man
woman, woman
woman, woman
man, man
man, woman
woman, woman

2-Class Confusion Matrix

- Let’s start off and calculate the classification accuracy for this set of predictions.
- The algorithm made 7 of the 10 predictions correct with an accuracy of 70%.
- accuracy = total correct predictions / total predictions made * 100
- accuracy = 7 / 10 * 100
- But what type of errors were made?
- Let’s turn our results into a confusion matrix.
- First, we must calculate the number of correct predictions for each class.

2-Class Confusion Matrix

- men classified as men: 3
- women classified as women: 4

- We can now arrange these values into the 2-class confusion matrix:

	men	women	
men		3	1
women	2	4	

	Positive	**Negative**
Positive	True Positive	False Positive
Negative	False Negative	True Negative

- The total actual men in the dataset is the sum of the values on the men column (3 + 2)
- The total actual women in the dataset is the sum of values in the women column (1 +4).
- The correct values are organized in a diagonal line from top left to bottom-right of the matrix (3 + 4).
- More errors were made by predicting men as women than predicting women as men.

3 Class Confusion Matrix

- Sometimes it may be desirable to select a model with a lower accuracy because it has a greater predictive power on the problem.
- **Confusion Matrix in R with caret**

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
# Make predictions on validation dataset
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
print(predictions)
print("Accuracy Score :", accuracy_score(Y_validation, predictions))
print("Confusion Matrix : \n",confusion_matrix(Y_validation,
predictions))
```

```
Confusion Matrix : [[ 7 0 0]
                    [ 0 11 1]
                    [ 0 2 9]]
```

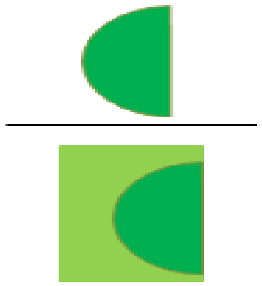
Prediction	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	7	0	0
Iris-versicolor	0	11	1
Iris-virginica	0	2	9

Metrics for Evaluating Performance of the Models

3. Recall Score Metric : out of all the positive examples there were, what fraction did the classifier pick up?

- Recall (also known as sensitivity) is the fraction of positives events that you predicted correctly as shown below:
- from sklearn.metrics import recall_score
- recall_score(df.actual_label.values, df.predicted_RF.values)

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN} =$$



Fraction of positives predicted correctly

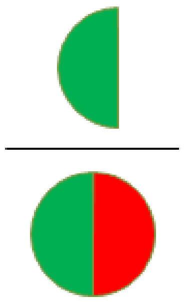
Metrics for Evaluating Performance of the Models

4. Precision Score Metric

precision answers the following question: out of all the examples the classifier labeled as positive, what fraction were correct?

- Precision is the fraction of predicted positives events that are actually positive as shown below:
- from sklearn.metrics import precision_score
- precision_score(df.actual_label.values, df.predicted_RF.values)

$$\text{Precision} = \frac{TP}{TP + FP} =$$



Fraction of predicted positives that are actually positive

Metrics for Evaluating Performance of the Models

5. F1 Score Metric

- The f1 score is the harmonic mean of recall and precision, with a higher score as a better model. The f1 score is calculated using the following formula:
- from sklearn.metrics import f1_score
- f1_score(df.actual_label.values, df.predicted_RF.values)

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * (precision * recall)}{precision + recall}$$

Classification Report

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
# Make predictions on validation dataset
knn = KNeighborsClassifier()
knn.fit(X_train, Y_train)
predictions = knn.predict(X_validation)
print(predictions)
print("Accuracy Score :", accuracy_score(Y_validation, predictions))
print("Classification Report :\n",classification_report(Y_validation, predictions))
```

```
Classification Report :
                  precision   recall  f1-score  support
Iris-setosa          1.00      1.00     1.00        7
Iris-versicolor      0.85      0.92     0.88       12
Iris-virginica       0.90      0.82     0.86       11
```

Conclusion

- If the classifier does not make mistakes, then precision = recall = 1.0. But difficult to achieve.
- In predictive analytics, when deciding between two models it is important to pick a single performance metric.
- As you can see here, there are many that you can choose from (accuracy, recall, precision, f1-score etc).
- Ultimately, you should use the performance metric that is most suitable for the business problem at hand.

Titanic Project

- https://www.ritchieng.com/machine-learning-project-titanic-survival/