Carlos J. Costa

EVALUATION

Evaluation

- After a data scientist has chosen a target variable and completed the prerequisites of transforming data and building a model, one of the final steps is evaluating the model’s performance.

Confusion Matrix

- This matrix describes an output of “yes” vs. “no”.
- These two outcomes are the “classes” of each example.

Predict

actual			
		No	yes
	No	90	10
	yes	5	95

Confusion Matrix

- To better interpret the table, it is possible to see it in terms of:
  - true positives (TP): number of positive records rightly predicted as positive
  - true negatives (TN): number of negatives records rightly predicted as negative
  - false positives (FP): number of negative records wrongly predicted as positive
  - false negatives (FN): number of positive records wrongly predicted as negative.

actual	Predict: No	Predict: yes
No	True Negative	False Positive
yes	False Negative	True Positive

Confusion Matrix

- False Positive is Type I Error
- False Negative is Type II Error

		Predict	
actual		No	yes
	No	True Negative	False Positive (Type I)
	yes	False Negative (Type II)	True Positive

Accuracy

- Overall performance of the model

Accuracy=(TP+TN)/(TP+FP+TN+FN)

or

Accuracy= All Correct/All

- Overall, how often is our model correct?

Predict

actual	No	yes
No	True Negative	False Positive
yes	False Negative	True Positive

Precision or positive predictive value (PPV)

- How accurate the positive predictions are
  - Precision=TP/(TP+FP)
  - or
  - Precision= True positives/predicted positives

- Precision helps when the costs of false positives are high.
  - e.g. detect skin cancer

Predict

actual	No	yes
No	True Negative	False Positive
yes	False Negative	True Positive

Recall or true positive rate (TPR)

- Coverage of actual positive sample

$$Recall=TP/(TP+FN)$$

- Recall helps when the cost of false negatives is high.
  - e.g. detect nuclear missiles

Predict

actual	No	yes
No	True Negative	False Positive
yes	False Negative	True Positive

Specificity or true negative rate (TNR)

- Coverage of Actual negative Sample
- Negative predictive value

Specificity=TN/(TN+FP)

actual	Predict: No	Predict: yes
No	True Negative	False Positive
yes	False Negative	True Positive

F1 Score

- Hybrid metric useful for unbalanced samples

$$F1=2((precision \times recall)/(precision + recall))$$

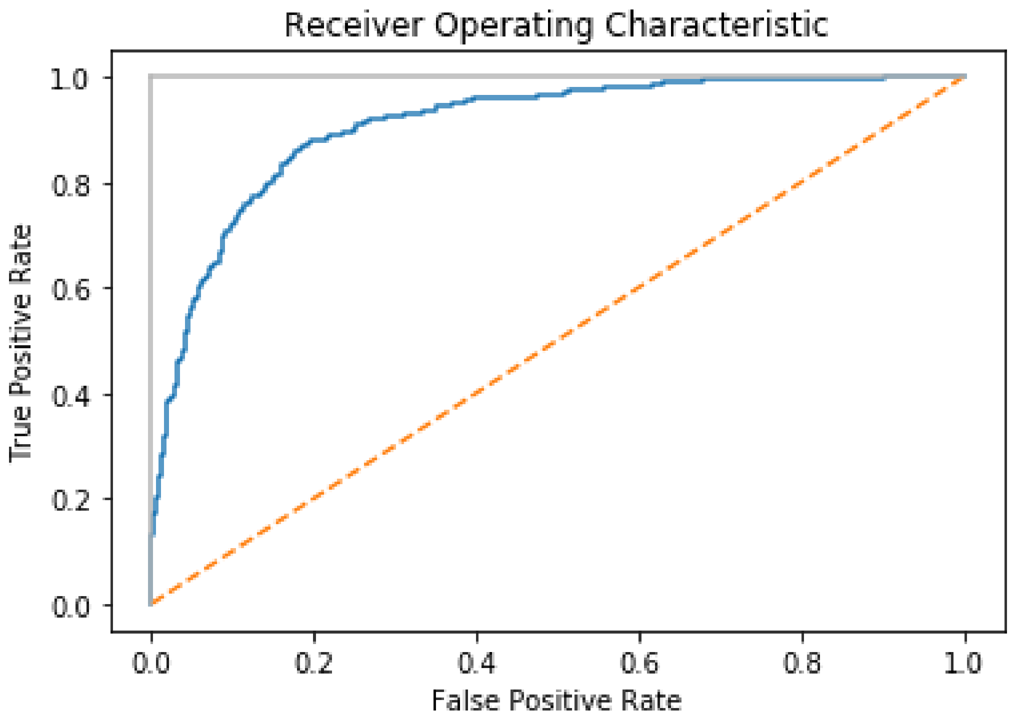
- a good F1 score means:
  - low false positives &
  - low false negatives
- correctly identifying real threats
- not disturbed by false alarms.

	Predict	
actual	No	yes
No	True Negative	False Positive
yes	False Negative	True Positive

ROC

- Receiver operating characteristic curve
- Specially useful in presence of binary non balanced datasets.
- ROC Charts present the balance between True Positive rate (recall) and False Positive rate in a graphical way,
- ROC Charts are available through the roc_curve method in the sklearn.metrics

ROC


Receiver Operating Characteristic
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

Python

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
#Accuracy
accuracy_score(y_test, y_pred)
# Precision
precision_score(y_test, y_pred)
#Recall
recall_score(y_test, y_pred)
# F1 Score
f1_score(y_test, y_pred)
```

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{P}+\mathrm{N}} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$$

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} = 1-\mathrm{FDR}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = 1-\mathrm{FNR}$$

$$F_1 = 2 \times \frac{\mathrm{PPV} \times \mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$$

Example Python: Import Data

```
import pandas as pd
import numpy as np
from sklearn import datasets
# Load the breast cancer data set
bc = datasets.load_breast_cancer()
X = bc.data
y = bc.target
```

Example Python: Split Sample

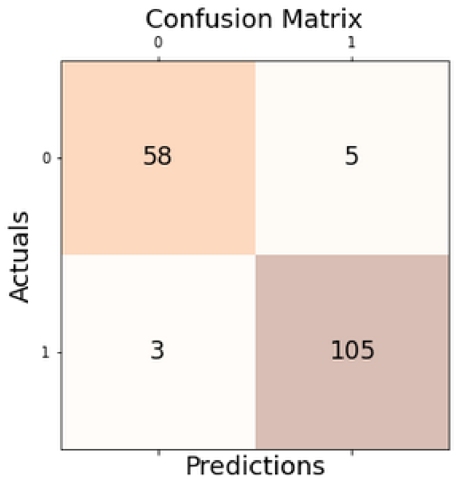
```
from sklearn.model_selection import train_test_split
# Create training and test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

Example Python: Fit Model and get predictions

```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
import matplotlib.pyplot as plt

# Standardize the data set
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

# Fit the SVC model
svc = SVC(kernel='linear', C=10.0, random_state=1)
svc.fit(X_train, y_train)

# Get the predictions
y_pred = svc.predict(X_test)
```

Example Python: Confusion matrix

```
# Calculate the confusion matrix
conf_matrix = confusion_matrix(y_true=y_test, y_pred=y_pred)
# Print the confusion matrix using Matplotlib
fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(conf_matrix, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```



Python: Show scores

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
#Accuracy
accuracy_score(y_test, y_pred)
# Precision
precision_score(y_test, y_pred)
#Recall
recall_score(y_test, y_pred)
# F1 Score
f1_score(y_test, y_pred)
```

Cross Validation

```
from sklearn import model_selection
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
#
#
models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
#
#
results = []
names = []
scoring = 'accuracy'
#scoring = 'recall'
seed = 7

for name, model in models:
    #, random_state=seed
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X, Y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

Regression

- **Mean absolute error**:
  - average of absolute errors of all the data points in the given dataset.
- **Mean squared error**:
  - average of the squares of the errors of all the data points in the given dataset.
  - one of the most popular metrics

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

- **Median absolute error**:
  - median of all the errors in the given dataset.
  - main advantage of this metric is that it's robust to outliers.
  - single bad point in the test dataset wouldn't skew the entire error metric, as opposed to a mean error metric.

Regression

- **Explained variance score**:

$$explained\ variance(y, \hat{y}) = 1 - \frac{Var(y - \hat{y})}{Var(y)}$$

  - measures how well our model can account for the variation in our dataset.
  - score of 1.0 indicates that our model is perfect.

- **R2 score (**R-squared **)**:

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

  - refers to the coefficient of determination.
  - how well the unknown samples will be predicted by our model.
  - the best possible score is 1.0, but the score can be negative as well.

Regression

```
import sklearn.metrics as sm
# Mean absolute error
sm.mean_absolute_error(y_test, y_test_pred)
# Mean squared error
sm.mean_squared_error(y_test, y_test_pred)
# Median absolute error
sm.median_absolute_error(y_test, y_test_pred)
# Explain variance score
sm.explained_variance_score(y_test, y_test_pred)
#R2 score
sm.r2_score(y_test, y_test_pred)
```