Evaluasi_Model_Klasifikasi

December 16, 2022

1 Model Evaluation

Evaluasi model, digunakan untuk mengetahui algoritma yang paling bagus berdasarkan dataset yang kita berikan untuk menyelesaikan masalah tertentu. Dalam istilah machine learning, disebut **Best Fit**. Evaluasi ini digunakan untuk mengukur kinerja berbagai model machine learning, berdasarkan kumpulan data masukan yang sama. Metode evaluasi berfokus pada akurasi model, dalam memprediksi hasil akhir.

Untuk mengevaluasi kinerja model Machine Learning, ada beberapa Metrik untuk mengetahui kinerjanya dan diterapkan untuk algoritma Regresi dan Klasifikasi. Berbagai jenis metrik klasifikasi adalah:

1. Confusion Matrix

2. Classification Accuracy
3. Precision
4. Recall
5. Specificity
6. F1- Score
7. Precision-Recall or PR curve
8. ROC (Receiver Operating Characteristics) curve
9. Area under Curve (AUC)

Confusion matrix

1. Accuracy Metrik yang paling umum digunakan untuk menilai model dan sebenarnya bukan merupakan indikator kinerja yang jelas. Dan Akan lebih buruk hasilnya jika data yang digunakan adalah imbalanced dataset.

[ ]:
```
# Importing necessary libraries
from sklearn.datasets import load_breast_cancer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

# Loading the breast cancer data set
diabetes_data = load_breast_cancer()
```

```
# Load dataset into dataframe
df_breast_cancer = pd.DataFrame(data=diabetes_data.data,
                                columns= diabetes_data.feature_names)

df_breast_cancer.head()
```

[ ]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	
1	0.07864	0.0869	0.07017	0.1812	
2	0.15990	0.1974	0.12790	0.2069	
3	0.28390	0.2414	0.10520	0.2597	
4	0.13280	0.1980	0.10430	0.1809	

	mean fractal dimension	…	worst radius	worst texture	worst perimeter	\
0	0.07871	…	25.38	17.33	184.60	
1	0.05667	…	24.99	23.41	158.80	
2	0.05999	…	23.57	25.53	152.50	
3	0.09744	…	14.91	26.50	98.87	
4	0.05883	…	22.54	16.67	152.20	

	worst area	worst smoothness	worst compactness	worst concavity	\
0	2019.0	0.1622	0.6656	0.7119	
1	1956.0	0.1238	0.1866	0.2416	
2	1709.0	0.1444	0.4245	0.4504	
3	567.7	0.2098	0.8663	0.6869	
4	1575.0	0.1374	0.2050	0.4000	

	worst concave points	worst symmetry	worst fractal dimension
0	0.2654	0.4601	0.11890
1	0.1860	0.2750	0.08902
2	0.2430	0.3613	0.08758
3	0.2575	0.6638	0.17300
4	0.1625	0.2364	0.07678

[5 rows x 30 columns]

```
# Creating independent and dependent variables
X = diabetes_data.data
y = diabetes_data.target
```

[ ]:

```
# Splitting the data into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=24)

print(f"Train Data: {X_train.shape}, {y_train.shape}")
print(f"Test Data: {X_test.shape}, {y_test.shape}")
```

```
Train Data: (455, 30), (455,)
Test Data: (114, 30), (114,)
```

[ ]:
```
# Create model Classifier
# Training a binary classifier using Random Forest Algorithm with default␣
 ↪hyperparameters
classifier = RandomForestClassifier(random_state=18)
classifier.fit(X_train, y_train)
```

[ ]:
```
RandomForestClassifier(random_state=18)
```

[ ]:
```
# Predict data using test data
# Here X_test, y_test are the test data points
predictions = classifier.predict(X_test)
```

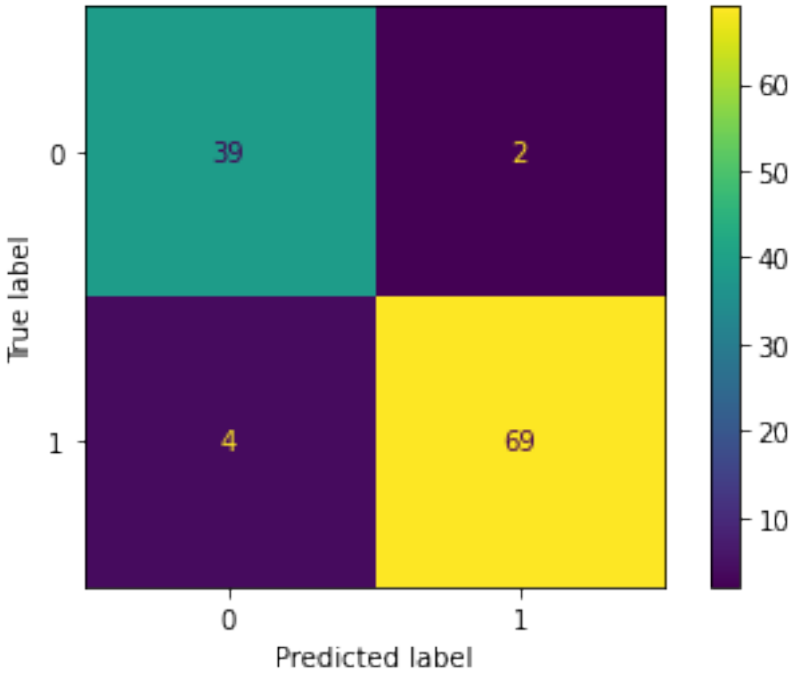
[ ]:
```
# importing all necessary libraries
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix

# confusion_matrix funnction a matrix containing the summary of predictions
print(confusion_matrix(y_test, predictions))

# plot_confusion_matrix function is used to visualize the confusion matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

```
[[39  2]
 [ 4 69]]
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



[ ]:
```
# Importing all necessary libraries
from sklearn.metrics import accuracy_score
# Calculating the accuracy of classifier
print(f"Accuracy of the classifier is: {accuracy_score(y_test, predictions)}")
```

```
Accuracy of the classifier is: 0.9473684210526315
```

[ ]:
```
TP = 69
TN = 39
FP = 2
FN = 4

acc = (TP+TN)/(TP+TN+FP+FN)
print(acc)
```

```
0.9473684210526315
```

2 3. Precision

Persentase kejadian positif dari total kejadian positif yang diprediksi. Di sini penyebut adalah prediksi model yang dilakukan sebagai positif dari seluruh kumpulan data yang diberikan. Ambillah untuk mengetahui 'seberapa banyak model yang benar ketika dikatakan benar'.

[ ]:
```
# Importing all necessary libraries
from sklearn.metrics import precision_score

# Calculating the precision score of classifier
print(f"Precision Score of the classifier is: {precision_score(y_test,␣
 ↪predictions)}")
```

```
Precision Score of the classifier is: 0.971830985915493
```

[ ]:
```
precision=TP/(TP+FP)
print(precision)
```

```
0.971830985915493
```

2.1 4. Recall/Sensitivity/True Positive Rate

Persentase kejadian positif dari total kejadian positif aktual. Oleh karena itu penyebut (TP + FN) di sini adalah jumlah sebenarnya dari kejadian positif yang ada dalam kumpulan data. Anggap saja untuk mencari tahu 'berapa banyak yang lebih tepat, modelnya terlewatkan ketika menunjukkan yang benar'.

[ ]:
```
# Importing all necessary libraries
from sklearn.metrics import recall_score

# Calculating the recall score of classifier
print(f"Recall Score of the classifier is: {recall_score(y_test, predictions)}")
```

```
Recall Score of the classifier is: 0.9452054794520548
```

[ ]:
```
recall=TP/(TP+FN)
print (recall)
```

```
0.9452054794520548
```

2.2 5. Specificity

Persentase kejadian negatif dari total kejadian negatif aktual. Oleh karena itu penyebut (TN + FP) di sini adalah jumlah sebenarnya dari instance negatif yang ada dalam kumpulan data. Ini mirip dengan recall tetapi pergeserannya ada pada contoh negatif. Seperti mencari tahu berapa banyak pasien sehat yang tidak mengidap kanker dan diberitahu bahwa mereka tidak mengidap kanker. Semacam ukuran untuk melihat seberapa terpisah kelas-kelas itu.

[ ]:
```
# Importing all necessary libraries
from imblearn.metrics import specificity_score
# Calculating the Specifity score of classifier
print(f"Specificity Score of the classifier is: {specificity_score(y_test,␣
 ↪predictions)}")
```

```
Specificity Score of the classifier is: 0.9512195121951219
```

[ ]:
```
specifity=TN/(TN+FP)
print(specifity)
```

```
0.9512195121951219
```

3 6. F1 Score

Ini adalah rata-rata Precision dan recall yang harmonis. Ini mengambil kontribusi keduanya, jadi semakin tinggi skor F1 semakin baik. Lihat bahwa karena produk di pembilang jika ada yang rendah, skor akhir F1 turun secara signifikan. Jadi model bekerja dengan baik dalam skor F1 jika prediksi positif sebenarnya positif (presisi) dan tidak melewatkan positif dan memprediksinya negatif (recall).

[ ]:
```
# Importing all necessary libraries
from sklearn.metrics import f1_score

# Calculating the F1 score of classifier
print(f"F1 Score of the classifier is: {f1_score(y_test, predictions)}")
```

```
F1 Score of the classifier is: 0.9583333333333334
```

[ ]:
```
f1score=2*precision*recall/(precision+recall)
print(f1score)
```

```
0.9583333333333334
```

4 Menggunakan Classification Report dari Sklearn

- precision
- recall
- F1-score
- Accuration

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

print(classification_report(y_test,predictions,zero_division=0))
```

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93        41
           1       0.97      0.95      0.96        73

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.94       114
```

```
weighted avg       0.95      0.95      0.95       114
```

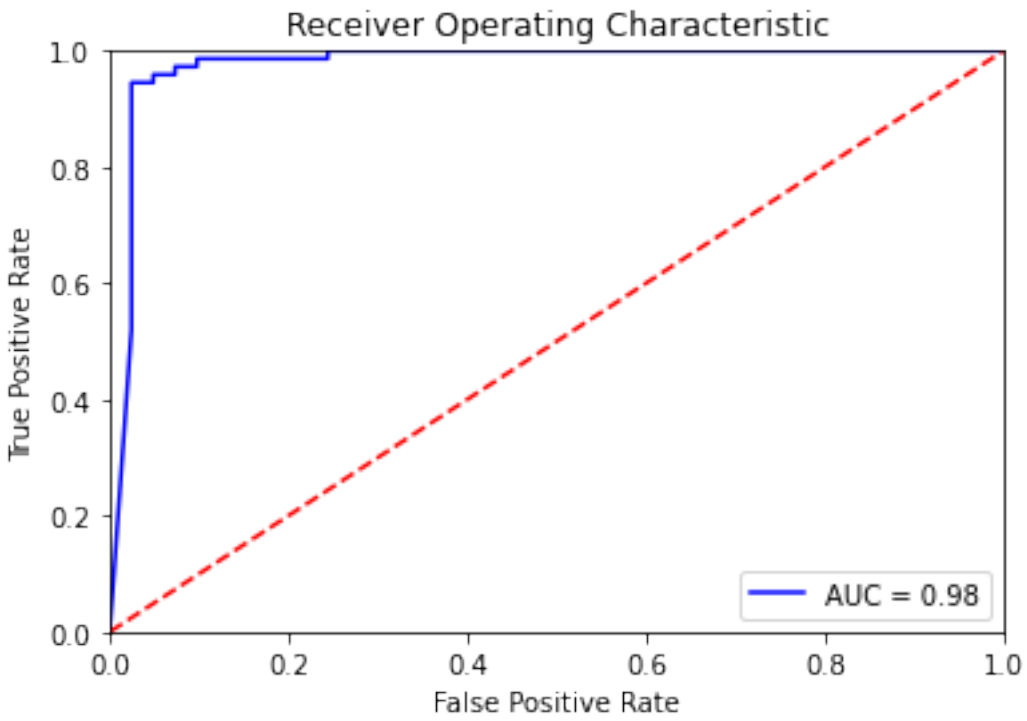
5 ROC

ROC merupakan singkatan dari (Receiver Operating Characteristic)dan digambarkan dalam grafik dari nilai TPR dan FPR.ROC adalah Metrik Evaluasi yang banyak digunakan, terutama digunakan untuk Klasifikasi **Biner**. True Positive Rate(TPR) dan False Positive Rate(FPR) memiliki nilai mulai dari 0 hingga 1. TPR dan FPR dihitung dengan nilai Treshold yang berbeda2 dan grafik digambar untuk lebih memahami tentang data. ROC adalah kurva probabilitas dan AUC mewakili derajat atau ukuran keterpisahan. Ini memberi tahu berapa banyak model yang mampu membedakan antar kelas. Dengan demikian, Area Di Bawah Kurva adalah plot antara False Positive Rate(FPR) dan True Positive rate pada nilai [0,1] yang berbeda. ROC AUC hanyalah area di bawah kurva, semakin tinggi nilai numeriknya semakin baik.

[ ]:
```python
# Importing all necessary libraries
from sklearn.metrics import roc_curve, auc
class_probabilities = classifier.predict_proba(X_test)
preds = class_probabilities[:, 1]
fpr, tpr, threshold = roc_curve(y_test, preds)
roc_auc = auc(fpr, tpr)
# Printing AUC
print(f"AUC for our classifier is: {roc_auc}")

# Plotting the ROC
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

```
AUC for our classifier is: 0.9769462078182426
```



Referensi:

1. https://towardsdatascience.com/various-ways-to-evaluate-a-machine-learning-models-performance-230449055f15
2. https://medium.com/analytics-vidhya/evaluation-metrics-for-classification-problems-with-implementation-in-python-a20193b4f2c3
3. https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826