Nama : Ghina Khoerunnisa
Program : Data Scientist Intern

Classification I

Classification merupakan salah satu kasus di supervised learning dimana prediksinya berupa class atau kategori dari suatu entitas berdasarkan fitur-fiturnya. Terdapat dua jenis masalah classification yaitu binary/binomial classification dan multiclass atau multinomial classification. Contoh classification adalah spam detection, image recognition, sentiment analysis, klasifikasi biologis, dll.

Logistic Regression bekerja dengan memprediksi probabilitas sampel yang kita punya termasuk ke dalam satu klasifikasi atau klasifikasi lainnya. Logistic Regression bagus digunakan untuk menyelesaikan permasalahan binary classification karena beroperasi pada probabilitas. Berdasarkan probabilitas tersebut akan diberikan klasifikasi 1 atau 0. Pada dasarnya Logistic Regression membulatkan nilai untuk memberikan klasifikasi. 0 adalah kelas negatif dan 1 adalah kelas positif. Tipe-tipe dari logistic regression adalah Binary (dua kemungkinan hasil), Multinomial (lebih dari dua kemungkinan hasilnya dan tidak berurutan), dan Ordinal (lebih dari dua kemungkinan hasilnya dan berurutan). Logistic Regression juga menggunakan persamaan linear (mirip dengan linear regression) tetapi persamaan linearnya ini dimasukkan ke dalam sigmoid function.

$$y' = \text{sigmoid}(w1.x1 + \ldots + wn.\ xn + b)$$

Dan cost function nya adalah,

$$ce = y*\log(y') + (1-y)*\log(1-y')$$

Single Variate Logistic Regression adalah kasus logistic regression dimana terdapat hanya satu variabel independen. Sedangkan multivariate logistic regression adalah kasus logistic regression dimana variabel independennya lebih dari satu. Untuk menggunakan logistic regression di python dapat memakai library scikit learn LogisticRegression.

K-Nearest Neighbor (KNN), KNN merupakan algoritma yang bekerja dengan K tetangga terdekat. Jika ada suatu titik P maka kita dapat menghitung jarak antar titik lainnya ke titik P kemudian mengambil observasi sebanyak K tetangga terdekatnya misalnya 7 tetangga berarti ada 7 titik terdekat lainnya dari titik P. Kemudian dari 7 titik ini voting class/kategori apa yang dominan, class dominan ini lah yang akan menentukan class dari titik P tersebut. Untuk mengukur jaraknya kita dapat menggunakan euclidean, hamming, manhatta, minkowski distance tetapi biasanya yang digunakan adalah euclidean.

Confusion Matrix adalah pengukuran performa untuk masalah klasifikasi machine learning. Terdapat 4 kombinasi antara predicted value dan actual value. TN (predicted 0, actual 0), FP (predicted 1, actual 0), FN (predicted 0, actual 1), TP (predicted 1, actual 1).

Classification Report merupakan report yang menampilkan score precision, recall, f1-score untuk setiap class. Precision merupakan hasil bagi antara TP dengan jumlah semua prediksi positif (FP + TP). Recall merupakan hasil bagi antara TP dengan FN+TP. Menentukan performansi yang harus ditingkatkan tergantung dengan situasi kasusnya. Jika kita ingin memperkecil FN maka kita perlu meningkatkan recall sedangkan ketika kita ingin memperkecil FP maka kita perlu meningkatkan precision.

Implementasi

- **Implementasi dengan Logistic Regression**

```
model = LogisticRegression(solver='liblinear', C=0.05, multi_class='ovr',
                           random_state=0)
model.fit(x_train, y_train)
```

```
LogisticRegression(C=0.05, multi_class='ovr', random_state=0,
                   solver='liblinear')
```

```
y_pred = model.predict(x_test)
```

```
model.score(x_train, y_train)
```

```
0.964509394572025
```

```
confusion_matrix(y_test, y_pred)
```

```
array([[27,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 32,  0,  0,  0,  0,  1,  0,  1,  1],
       [ 1,  1, 33,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  1, 28,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0, 29,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 39,  0,  0,  0,  1],
       [ 0,  1,  0,  0,  0,  0, 43,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 39,  0,  0],
       [ 0,  2,  1,  2,  0,  0,  0,  1, 33,  0],
       [ 0,  0,  0,  1,  0,  1,  0,  2,  1, 36]], dtype=int64)
```

```
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.96	1.00	0.98	27
1	0.89	0.91	0.90	35
2	0.94	0.92	0.93	36
3	0.88	0.97	0.92	29
4	1.00	0.97	0.98	30
5	0.97	0.97	0.97	40
6	0.98	0.98	0.98	44
7	0.91	1.00	0.95	39
8	0.94	0.85	0.89	39
9	0.95	0.88	0.91	41
accuracy			0.94	360
macro avg	0.94	0.94	0.94	360
weighted avg	0.94	0.94	0.94	360

- **Implementasi dengan KNN**

```
knn_model = KNeighborsClassifier(n_neighbors=7)
knn_model.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=7)
```

```
y_pred_knn = knn_model.predict(X_test)
```

```
knn_model.score(X_train, y_train)
```

```
0.97625
```

```
confusion_matrix(y_test, y_pred_knn)
```

```
array([[504,  14],
       [ 23, 460]], dtype=int64)
```

```
print(classification_report(y_test, y_pred_knn))
```

	precision	recall	f1-score	support
0	0.96	0.97	0.96	518
1	0.97	0.95	0.96	483
accuracy			0.96	1001
macro avg	0.96	0.96	0.96	1001
weighted avg	0.96	0.96	0.96	1001