**EVALUASI MODEL**

**Pendahuluan**

Proses evaluasi model yang telah kita buat merupakan hal penting dalam mengembangkan model *Machine Learning* yang baik. Pada bagian ini akan dibahas evaluasi model khususnya untuk klasifikasi. Proses evaluasi model dapat dilakukan setelah *training model* selesai. Evaluasi model menggunakan data evaluasi yang tidak boleh sama dengan data yang digunakan untuk men-training model. Pengujian dengan data evaluasi ini akan menghasilkan nilai akurasi sesungguhnya dari sebuah model yang sudah ditraining. Tetapi, akurasi bukanlah satu-satunya nilai yang diperhatikan dalam melakukan evaluasi karena bisa saja nilai akurasi yang tinggi menipu akibat dari ketidakseimbangan dataset. Oleh karena itu diperlukan metrik evaluasi lain yang lebih komprehensif, seperti Confusion Matrix, Precision, dan Recall.

**Akurasi**

Secara sederhana, akurasi dapat diperoleh dengan melakukan perbandingan banyaknya jumlah prediksi yang tepat terhadap sejumlah prediksi yang dilakukan dan dinyatakan dalam persentase.

𝑡

𝑎 = × 100%

𝑛

dengan *a* adalah akurasi dalam persen, *t* adalah jumlah percobaan prediksi benar, dan *n* adalah jumlah percobaan.

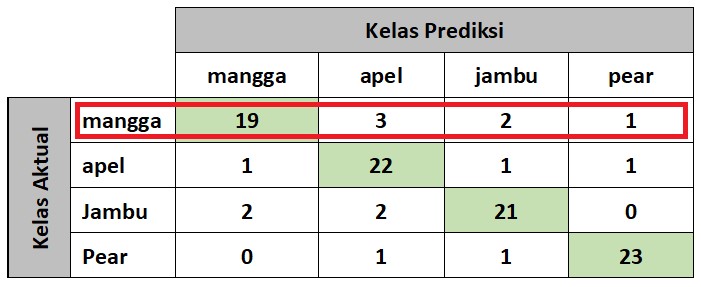
Untuk evaluasi awal, akurasi dengan pengukuran di atas dapat digunakan sebagai evaluasi model. Evaluasi lebih lanjut seringkali diperlukan untuk menilai performansi dengan lebih mendalam. Misalnya, pada pengujian 100 orang yang diduga Covid-19 dengan suatu alat adalah 90% yang tampakanya cukup tinggi untuk sebuah alat pendeteksi baru. Data yang sesungguhnya adalah 90 orang negatif dan 10 orang positif dan hasil pengujian dengan alat baru tersebut adalah 100 orang tersebut dinyatakan negatif. Angka 90 persen menjadi tidak teralu berarti lagi karena kesalahan yang walaupun hanya 10% namun merupakan kesalahan fatal. Evaluasi model yang lain yang lebih komprehensif dilakukan dengan memperhatikan lebih detil data hasil pengujian untuk masing-masing kelas yang akan dibahas berikutnya.

**Confusion Matrix**

Confucion matrix merupakan tabulasi detil yang menggambarkan hasil klasifikasi pada suatu model. Untuk setiap percobaan, hasil klasifikasi dicatat dengan memperhatikan kelas yang sesungguhnya dengan hasil percobaan yang dilakukan. Berikut ini merupakan contoh confusion matrix untuk klasifikasi citra buah buahan, dalam hal ini buah mangga, apel, jambu, dan pear.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | **Kelas Prediksi** | |  |
| **mangga** | **apel** | **jambu** | **pear** |
| **Kelas**  **Aktual** | **mangga** | **19** | **3** | **2** | **1** |
| **apel** | **1** | **22** | **1** | **1** |
| **Jambu** | **2** | **2** | **21** | **0** |
| **Pear** | **0** | **1** | **1** | **23** |

Pada contoh tersebut, pengujian terhadap citra mangga dilakukan 25 kali terhadap sekumpulan data set citra mangga, dimana satu citra mangga tepat diuji satu kali. Perhatikan area dalam kotak merah pada tabel berikut.



Pada area kotak merah, artinya 25 citra mangga yang diujikan terhadap model alogoritma klasifikasi berhasil mengidentifikasi 19 citra mangga secara benar, yaitu 19 citra mangga diidentifikasi sebagai citra mangga. Selanjutnya teradapat kesalahan dalam klasifikasi yaitu 3 citra mangga diidentifikasi sebagai citra apel, 2 citra mangga diidentifikasi sebagai citra jambu, dan 1 buah citra mangga diidentifkasi sebagai citra pear. Pada baris tersebut total percobaan adalah 25 kali dengan 19 percobaan memberikan hasil yang benar dan 6 percobaan memberikan hasil yang salah. Total percobaan pada keseluruhan buah di atas adalah 100 kali, dengan distribusi citra uji untuk masing-masing buah sebanyak 25 citra uji.

Tabulasi melalui *confision matrix* dapat menyajikan akurasi secara lebih baik, walaupun hasil akurasi akhir tetap berdasarkan kepada banyaknya percobaan valid terhadap total percobaan yang dilakukan. Hasil klasifikasi yang benar untuk tiap kelas berada tepat di diagonal matriks. Sedangkan distribusi kesalahan klasifikasi dapat dilihat di bagian selain diagonal.

**Binary Classification Problem**

Dalam klasifikasi, sering dijumpai kasus yang memiliki dua kelas saja. Karena hanya dua kelas maka kasus ini disebut klasifikasi biner (binary classification). Misalnya klasifikasi pada citra CT scan paru-paru pasien pada pengenalan penyakit Covid-19. Kasus ini hanya memperediksi dua kelas saja yaitu pasien diprediksi menderita Covid-19 (positif Covid19) atau tidak menderita Covid-19 (negatif Covid-19). Klasifikasi semacam ini secara sederhana akan mengkategorisasikan data yang diuji kedalam salah satu dari : kasus positif atau negatif. Kasus serupa dapat berupa jawaban terhadap klasifikasi : Yes/No, Bagus/Tidak Bagus, Enak/Tidak Enak, Matang/Tidak Matang, dan sebagainya. Contoh lain dari model biner adalah pengenalah sebuah objek saja dari kemungkinan banyak objek. Misalanya klasifikasi citra binatang untuk dikategorikan sebagai citra kucing atau bukan kucing. Pada kasus terakhir, secara sederhana hasil prediksi dari model akan menghasilkan keputusan apakah sebuah citra binatang merupakan citra kucing atau bukan. Pemilihan model pada kasus terakhir juga akan memaksa citra apapun akan diklasifikasi ke salah satu kelas dari kelas citra kucing atau kelas bukan kucing. Oleh karena itu, perlu diperhatikan asumsi awal pada model semacam ini, misalnya citra khusus untuk binatang kucing dan anjing saja, dengan fokus utama klasifikasi adalah mengidentifikasi kucing. Dengan mengambil fokus binatang kucing, maka prediksi postif berarti kucing sedangkan negatif adalah bukan kucing.

Pada binary classification, terdapat empat parameter yang dapat diperhatikan untuk mengevaluasi hasil prediksi dari sebuah model. Empat parameter tersebut adalah:

* True Positive (TP): nilai sesungguhnya adalah positif dan diprediksi postif
* False Positive (FP): nilai sesungguhnya adalah negatif namun diprediksi positif • True Negative (TN): nilai sesungguhnya adalah negatif dan diprediksi negatif, dan
* False Negative (FN): nilai sesungguhnya adalah positif namun diprediksi negatif.

Pada kasus klasifikasi citra kucing/bukan kucing di atas (perhatikan fokus utama klasifikasinya, yaitu kucing sehingga positif adalah kucing dan negatif adalah anjing): True Positive adalah ketika citra yang diuji merupakan foto kucing, kemudian hasil prediksi model adalah kucing juga. False Positive (FP) adalah ketika citra yang diuji merupakan foto anjing, tetapi hasil prediksi model adalah kucing. True Negative adalah ketika citra yang diuji merupakan foto anjing, kemudian hasil prediksi model adalah anjing juga. False Negative (FN) adalah ketika gambar masukkan merupakan foto kucing, tetapi hasil prediksi model adalah anjing.

**Confusion Matrix Pada Binary Classification**

Confusion Matrix merepresentasikan prediksi dengan kondisi sebenarnya dari data yang dihasilkan oleh model yang telah ditraining. Confusion Matrix akan menggunakan parameter TP, FP, TN, dan FN dalam representasinya yang secara sederhana apabila dinyatakan dalam tabel confusion matrix biner adalah sebagai berikut.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nilai Prediksi | |
| Positive | Negative |
| Nilai Aktual | Positive | True Positive | False Negative |
| Negative | False Positive | True Negative |

Pada kasus prediksi pengenalah Covid-19, kasus False Negative merupakan kasus yang laing tidak diharapkan. False Negative berarti seseorang sebenarnya menderita Covid-19 namun hasil pengujian menunjukkan negatif. Pada prediksi penyakit dan kebencanaan, umumnya False Negative merupakan kasus yang paling dihindari (paling ditekan sekecil mungkin) karena biasanya menyebabkan kerugian besar.

Berdasarkan Confusion Matrix, kita dapat menghitung nilai Precision, Recall, dan beberapa pengukuran lain sebagai berikut.

**Precision**

Precision adalah rasio dari data positif yang diklasifikasikan bernilai benar terhadap jumlah total hasil prediksi positif. Berikut adalah persamaan untuk mendapatkan nilai Precision

𝑇𝑃 𝑇𝑃 + 𝐹𝑃

𝑝𝑟𝑒𝑐𝑖𝑠𝑠𝑖𝑜𝑛 = .

**Recall**

Recall sering disebut juga sebagai **True Positive Rate** atau **Sensitivity**. Recall dapat didefinisikan sebagai rasio dari jumlah data positif yang diprediksi dengan benar (TP) terhadap jumlah data yang data secara aktual yang bernilai positif. High Recall menunjukkan kelas dikenali dengan baik (FN rendah), beriku adalah persamaan untuk mendapkan nilai Recall

𝑇𝑃 𝑇𝑃 + 𝐹𝑁

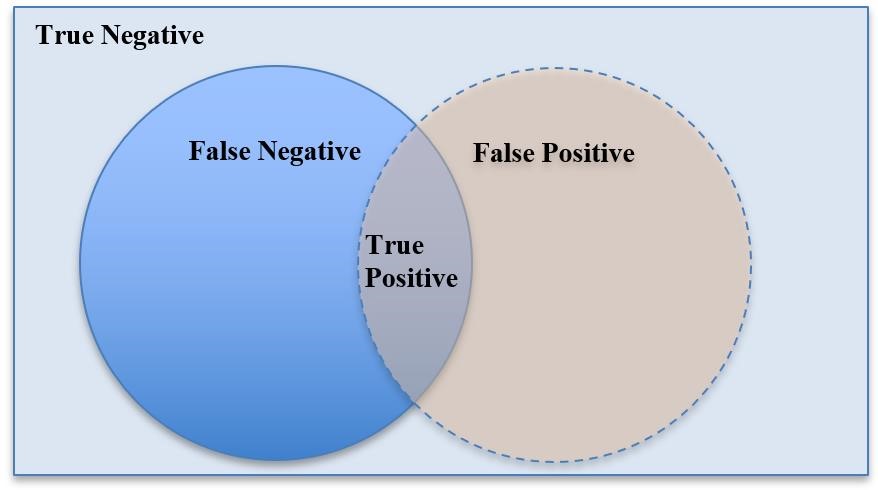
𝑟𝑒𝑐𝑎𝑙𝑙 = .

**Relasi Precision dengan Recall**

Nilai Precision dan Recall memiliki makna tentang bagaimana kemampuan memprediksi dari model yang telah kita buat dengan data training yang telah diberikan, sebagai evaluasi untuk meningkatkan kemampuan model sudah dibuat. Berikut beberapa contoh kondisinya:

**A. Low Recall, Low Precision**

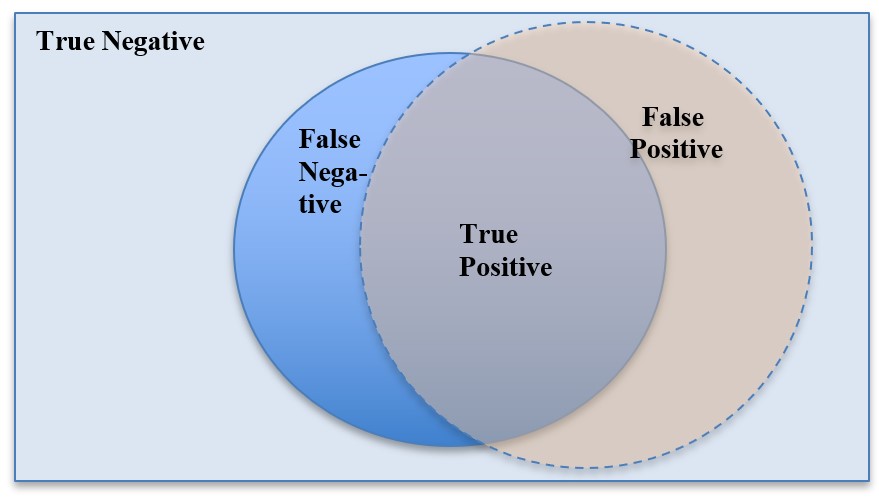
Model yang low recall dan low precision merupakan model yang memiliki kinerja kurang baik. Baik False Negative (FN) maupun False Positive (FP), yang keduanya merupakan prediksi dengan hasil yang salah, memiliki nilai yang besar. Model semacan ini dapat digambarkan melalui diagram sebagai berikut.



Pada kasus klasifikasi kucing di atas, model seperti ini menunjukkan bahwa banyak citra yang seharusnya kucing diidentifikasi sebagai anjing dan sebaliknya.

**B. High Recall, Low Precision**

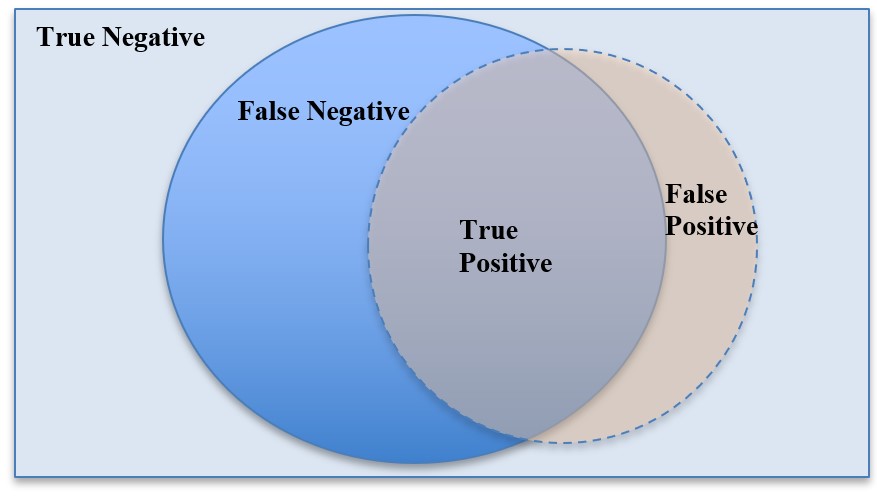
Kondisi memiliki makna bahwa sebagian besar data positif dapat dikenali dengan baik (FN Rendah), tetapi ada banyak data negatif dikenali sebagai positive (FP).



*Contoh kasus****:***Model untuk klasifikasi citra kucing dan bukan kucing. Model dapat mengklasifikasikan dengan benar banyak gambar kucing sebagai kucing tetapi banyak juga mengklasifikasikan anjing sebagai kucing.

**C. Low Recall, High Precision**

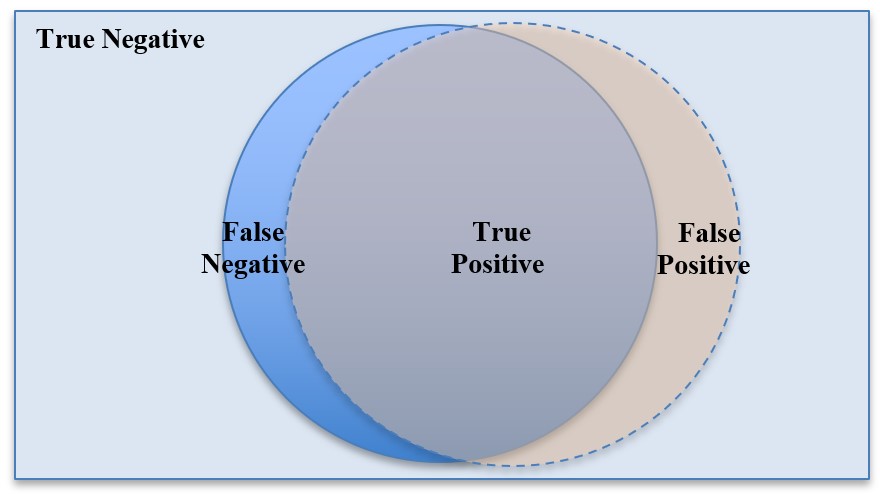
Model dapat mengklasifikasi lebih sedikit data bernilai positif (FN tinggi), tetapi model mengklasifikasikan data positif yang bernilai positif benar-benar bernilai positif (FP rendah).



*Contoh kasus:* Model untuk klasifikasi citra kucing dan bukan kucing.Hasil prediksi model berupa citra kucing, mayoritas adalah benar sebagai citra kucing. Sayangnya, banyak juga citra yang sebenarnya citra kucing namun diprediksi sebagai anjing.

**D. High Recall, High Precision**

Model yang diharapakan adalah model yang memiliki Recall maupun Precision yang tinggi. Kesalahan prediksi baik False Positive maupum False Negative dapat ditekan seminimal mungkin.



**F1-Score**

F1-Score menghitung hubungan antara Precision dan Recal yang dinyatakan dengan rumus

𝑝𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 • 𝑟𝑒𝑐𝑎𝑙𝑙

F1𝑆𝑐𝑜𝑟𝑒 = 2

𝑝𝑟𝑒𝑐𝑖𝑠𝑖𝑜𝑛 + 𝑟𝑒𝑐𝑎𝑙𝑙 .

**Specificity**

Specificity dikenal juga sebagai True Negative Rate. Specifity merupakan rasio data yang diprediksi negatif dengan benar terhadap data aktual yang negatif.

𝑇𝑁 𝑇𝑁 + 𝐹𝑃

𝑆𝑝𝑒𝑐𝑖𝑓𝑖𝑐𝑖𝑡𝑦 = .

**Negative Predictive Value**

Negative Predictive Value merupakan rasio data yang diprediksi negatif dengan benar terhadap data yang diprediksi negatif.

𝑇𝑁 𝑇𝑁 + 𝐹𝑁

𝑁𝑒𝑔𝑎𝑡𝑖𝑣𝑒 𝑃𝑟𝑒𝑑𝑖𝑐𝑡𝑖𝑣𝑒 𝑉𝑎𝑙𝑢𝑒 = .

**Accuracy**

Pada akhirnya, pengukuran model yang paling mendasar tetap diperlukan yaitu akurasi yang pada kasus klasifikasi biner dapat dirumuskan sebagai:

𝑇𝑃 + 𝑇𝑁 𝑇𝑃 + 𝑇𝑁 + 𝐹𝑃 + 𝐹𝑁

𝐴𝑐𝑐𝑢𝑟𝑎𝑐𝑦 = .

**Peta Evaluasi Model untuk Klasifikasi Biner**

Berbagai pengukuran performansi pada klasifikasi biner di atas dapat diintisarikan kedalam tabel berikut.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Nilai Prediksi** | |  |
| **Positive** | **Negative** |
| **Nilai**  **Aktual** | **Positive** | **True Positive (TP)** | **False Negative (FN)** | **Recall, Sensitivity,**  **True Positive Rate**  𝑇𝑃    𝑇𝑃 + 𝐹𝑁 |
| **Negative** | **False Positive(FP)** | **True Negative (TN)** | **Specificity, True Negative Rate**  𝑇𝑁    𝐹𝑃+𝑇𝑁 **False Positive Rate**  𝐹𝑃    𝐹𝑃+𝑇𝑁 |
|  |  | **Precision**    𝑇𝑃  .  𝑇𝑃 + 𝐹𝑃 | **Negative Predictive**  **Value**  𝑇𝑁  .  𝑇𝑁 + 𝐹𝑁 | **Accuracy**    𝑇𝑃 + 𝑇𝑁    𝑇𝑃 + 𝑇𝑁 + 𝐹𝑃 + 𝐹𝑁 |

Pada latihan berikut, akan digunakan Precison dan Recall sebagai sarana latihan dalam mengukur performansi model.

**Evaluasi Model untuk Probabilistik Model**

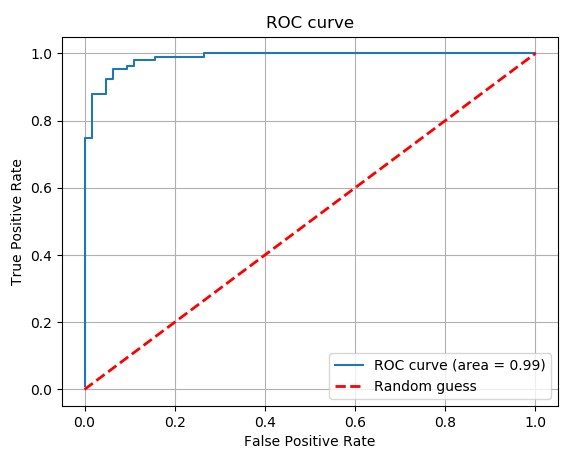
Hasil akhir sebuah model dalam memprediksi dengan menghasilkan nilai probabilitas [0,1] untuk suatu label , tetapi sebelum menjadi bentuk biner sebuah keluaran dari model memiliki nilia probabilitas, kemudian terbesar dari probabilitas menjadi 1 dan yang terkecil menjadi 0. Nilai Probabilitas ini menunujukkan seberapa yakin hasil prediksi model terhadap suatu kelas/label. Label ditentukan dengan menetapkan suatu threshold. Evaluasi model pun dapat dilakukan dalam sebuah model dengan memperhatikan nilai secara probabilistiknya, berikut beberapa metode diantaranya:

1. Receiver Operator Characterisic (ROC)
2. Logarithmic loss function

a. Receiver Operator Characterisic (ROC)

Evaluasi dapat dilakukan metode Receiver Operating Characterisic (ROC) yaitu dengan mem-plot antara Recall (atau disebut juga True Positive Rate) sebagai sumbu-y dengan False Positive Rate sebagai sumbu-x untuk setiap threshold klasifikasi yang mungkin (threshold antara 0 hingga 1). Area yang diperoleh dari ROC dapat digunakan untuk analisis ROC – AUC (AUC: Area Under Curve). Perhitungan nilai ROC-AUC diperlukan dalam evaluasi model karena nilai ROC-AUC dan grafik ROC-AUC lebih menggambarkan secara lengkap visualisasi untuk semua threshold klasifikasi yang ada.

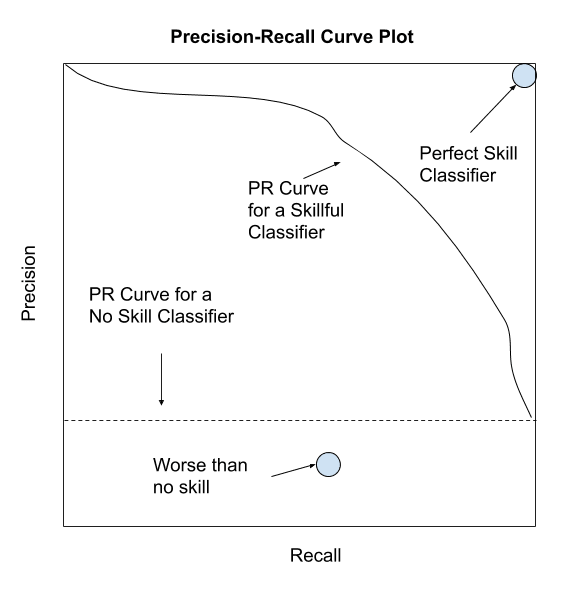
Sedangkan, Akurasi hanya merepresentasikan performansi pada satu nilai threshold. Hasil paling optimal adalah yang menghasilkan AUC paling luas



Source [: https://www.analyticsvidhya.com/blog/2020/10/quick-guide-to-evaluation-metrics-for-supervised-and-](https://www.analyticsvidhya.com/blog/2020/10/quick-guide-to-evaluation-metrics-for-supervised-and-unsupervised-machine-learning/)

[unsupervised-machine-learning/](https://www.analyticsvidhya.com/blog/2020/10/quick-guide-to-evaluation-metrics-for-supervised-and-unsupervised-machine-learning/)

Terdapat alternatif lain untuk mengevaluasi model dengan menggunakan ROC, yaitu Precision Recall (PR) - ROC. Untuk mendapatkan grafik PR-ROC dengan mem-plot Precision sebagai sumbu-y dan Recall sebagai sumbu-x. PR-ROC biasanya digunakan untuk melakukan evaluasi untuk kelas yang minoritas yang menjadi perhatian utama cukup kecil.

 Source : <https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/>

b. Logarithmic loss (log loss)

Logarithmic loss atau log loss menunjukkan seberapa yakin pemberian label terhadap data yang diuji/diobeservasi untuk setiap sampel dengan menghitung probabilitas untuk semua label yang mungkin terjadi. Nilai log loss berada di [0,~) dan dinyatakan sebagai:

−1 *N M*

*Logloss*= *N* *i*=1 *j*=1 *yij* log(*pij* ) ,

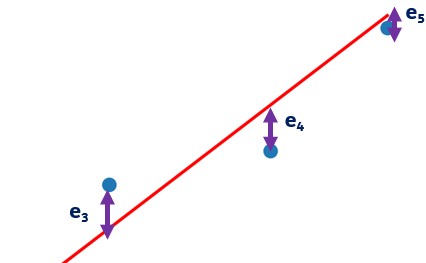
dengan N = jumlah sampel, M = jumlah kelas, *yij* = sampel, dan *pij* = probabilitas sampel.

Penerapan evaluasi model dapat digunakan untuk berbagai macam model, seperti pada bidang medis. Pada bidang medis seperti: deteksi penyakit menular (misalnya alat detektor G-Nose untuk penyakit Covid-19), deteksi kanker (misalnya kanker ganas / jinak pada kanker payudara , kanker serviks, dsb), deteksi kehamilan, dan deteksi kekurangan gizi. Adapun juga untuk bidang kebencanaan seperti: deteksi gunung meletus, deteksi kemarau panjang, deteksi banjir, deteksi gempa bumi, dan deteksi tsunami.

**Evaluasi Model untuk Regresi**

Evaluasi model tidak hanya dapat diguanakan pada Model berbasis klasifikasi, tetapi dapat juga untuk Model Regresi. Model Regresi adalah sebuah model yang memprediksi suatu nilai konitnyu (bilangan real) bukan nilai diskrit (berupa kelas/label). Contoh: prediksi harga rumah, suhu maksimum, kekuatan gempa, dan harga saham. Hasil prediksi dari Model Regresi memiliki nilai error yang merupakan selisih dari nilai aktual dengan nilai prediksi (real). Berikut merupakan contoh metode untuk mengevaluasi Model Regresi:

1. Mean Absolute Error (MAE)
2. Mean Squared Error (MSE)
3. Root Mean Squared Error (RMSE)
4. Mean Absolute Percentage Error (MAPE)
5. Mean Percentage Error (MPE)



a. Mean Absolute Error (MAE)

MAE adalah sebuah nilai yang didapatkan dari setiap selisih error diambil nilai mutlaknya untuk selanjutnya dijumlahkan. Jumlah nilai mutlak semua error dibagi rata dengan banyaknya sampel sehingga diperoleh nilai rata-rata error, karenanya disebut Mean Absolute Error, dinyatakan sebagai

1 *n*

*MAE* = *n* *j*=1| *y j* − *y*ˆ *j* |,

dengan *n* adalah banyak sampel, *y j* adalah nilai aktual untuk sampel j, dan *y*ˆ *j* adalah nilai prediksi untuk sampel j.

b. Root Mean Squared Error (RMSE)

RMSE adalah sebuah nilai yang didapatkan dari setiap selisih error diambil nilai kuadratnya untuk selanjutnya dijumlahkan. Jumlah nilai kuadrat setiap error dibagi rata dengan banyaknya sampel sehingga diperoleh nilai rata rata kuadrat error, untuk kemudian ditarik nilai akarnya. Berikut persamaan untuk RMSE

*RMSE*= 1*n**jn*=1(*y j* −*y*ˆ *j* )2 ,

dengan *n* adalah banyak sampel, *y j* adalah nilai aktual untuk sampel j, dan *y*ˆ *j* adalah nilai prediksi untuk sampel j.

**Pengukurkan Performansi pada Clustering**

Clustering merupakan termasuk salah satu jenis unspervised learning, bentuk data tidak berlabel (tidak ada kelas). Tujuan dari mengelompokkan data yang mirip sedekat mungkin dan memisahkan data yang tidak mirip sejauh mungkin. Sama seperti supervised learning pada model klasifikasi dan regresi, clustering pun memiliki metode untuk melakukan evaluasi pada model yang telah dirancang. Contoh pengukuran kinerja untuk model clustering:

1. Silhouette Coefficient
2. Rand Index
3. Mutual Information
4. Calinski-Harabasz Index (C-H Index)
5. Davies-Bouldin Index
6. Dunn Index

a. Silhouette Coefficient

Silhouette Coefficient atau silhouette score adalah sebuah nilai yang akan menunjukkan tingkat baik buruknya sebuah model clustering. Silhouette Coefficient memiliki nilai diantara -1 dan +1. Nilai -1 mengindikasikan clustering yang tidak tepat, disekitar 0 mengindikasikan adanya overlaping clustering, dan +1 mengindikasikan clustering yang padat dan terpisah dengan baik. Persamaan berikut digunakan untuk mendapatkan nilai dari Silhouette Coefficient

*s*= *b*−*a* , max(*a*,*b*)

dengan s adalah Silhouette Coefficient, a adalah rata-rata jarak sebuah sampel dengan sampel lainnya di cluster yang sama, dan b adalah rata-rata jarak sebuah sampel dengan sampel lainnya di cluster tetangga terdekat.

**Pembandingan Model Hasil Perancangan**

Model yang telah dirancang diharapkan dapat memiliki akurasi yang terbaik. Oleh karena itu, apabila kita telah merancang berbagai mode diperlukan untuk melakukan perbandingan kinerja untuk setiap modelnya. Contoh:

Dua buah model, yaitu model 1 dan model 2, dilakukan pengujian terhadap 10 data dengan hasil seperti di samping dengan menghitung rasio jumlah percobaan valid terhadap jumlah percobaan. Hasil dari pengujian tersebut adalah

Akurasi model 1: 6/10= 60%

Akurasi model 2: 5/10 = 50%

Akurasi model yang lebih tinggi belum cukup untuk dapat diklaim bahwa model tersebut secara statisik signifikan berbeda dan lebih baik dari model lainnya. Untuk mendukung klaim bahwa model 1 lebih baik dari model 2 perlu pengujian secara statistik dengan membuat dua hipotesa yang berlawanan:

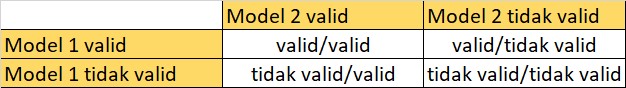
H0: Kedua model memiliki akurasi yang sama

H1: Kedua model memiliki akurasi yang berbeda

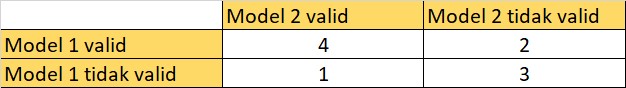
Pengujian statistik yang sederhana dapat dilakukan dengan McNemar’s Test.

**McNemar’s Test**

McNemar’s Test adalah salah satu metode evaluasi untuk membandingkan dua model secara statistik. Data pengujian akan disusun menjadi tabel contingency seagai berikut (perhatikan pasangan dalam model 1/model 2)



sehingga dari tabel sebelumnya diperoleh:



McNemar’s test statistic dihitung menggunakan persamaan

S = (valid/tidak valid - tidak valid/valid)2 / (valid / tidak valid + tidak valid/valid),

dengan S adalah klaim statistik konsen kepada perbedaan valid dan tidak valid pada kedua model, bukan pada akurasi maupun tingkat error. Melalui perhitungan statisik lebih lanjut, perlu memperhatikan masing masing nilai dalam tabel contingency. Distribusi *x*2 mengasumsikan nilai-nilai besar untuk nilai elemen-elemen pada tabel contingency. Untuk nilai kecil menggunakan distribusi Binomial. Secara praktikal nilai S melakukan proses koreksi. Untuk perhitungan detail statistik ini dapat dibaca di referensi.

Parameter penting dalam McNemar’s Test, selain s adalah p. Secara penggunaan praktikal, dapat digunakan perintah (dalam python) untuk mendapatkan dua nilai ini, dengan

memperhatikan apakah nilai elemen tabel contingency besar atau kecil Contoh:

Berdasarkan tabel contingency sebelumnya, dapat dituliskan sebagai berikut

T = [[4,2],

[1,3]],

Untuk case nilai-nilai kecil (misalnya tabel contingency T di atas), dapat digunakan perintah:

s, p = mcnemar(T, exact=True),

parameter lain adalah ambang batas p untuk threshold, yaitu α, misalnya α = 0.05. Berdasarkan nilai p dan ambang α dapat ditentukan:

Jika p > α, hipotesa H0 gagal untuk ditolak, kedua model scara statistik tidak ada perbedaan

Jika p < α, hipotesa H0 ditolak, kedua model scara statistik secara signifikan ada perbedaan McNemar’s adalah pengujian yang sederhana dan telah berkembang diantaranya 5xcv t-test beserta pengembangannya. Detil teori pengujian ini dapat dilihat di referensi.

**Program**

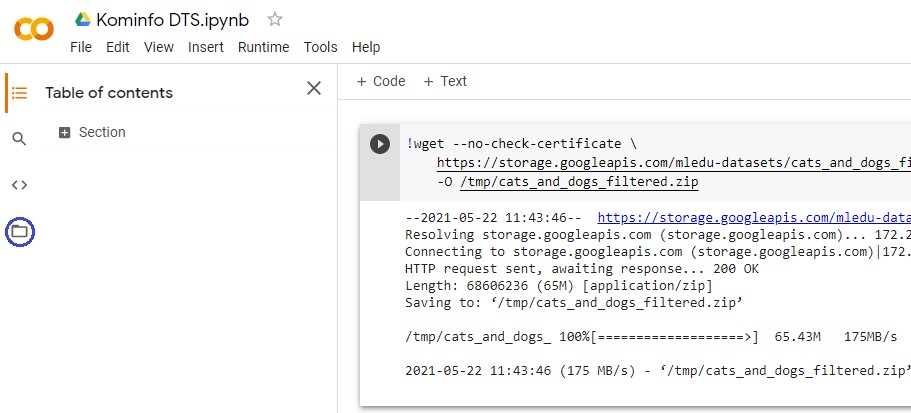
Pada modul ini, Google Colab digunakan untuk menjalankan program. Google Colab memiliki fleksibelitas tinggi sehingga saat ini Google Colab mulai banyak digunakan oleh para programmer khususnya yang menggunakan bahasa pemograman Python. Google Colab tidak jauh berbeda dengan Jupyter notebook, hanya saja untuk Google Colab Anda tidak perlu untuk melakukan instalasi sepanjang Anda memiliki akun Gmail. Anda dapat mengakses Google dimana saja selama terhubung dengan internet. Tidak hanya simpel dalam penggunaan saja, Google Colab juga memiliki kelebihan yaitu ketika Anda menjalankan program Anda di Google Colab, maka Anda menggunakan resource yang telah disediakan oleh Google Colab. Oleh karena itu, apabila komputer Anda tidak memiliki spek yang cukup bagus Anda dapat menggunakan Google untuk menggunakan spek super komputer dari Google. Hanya saja untuk layanan gratis Anda dapat menjalankan program dengan durasi 12 jam, apabila lebih dari 12 jam Anda perlu membayar. Seperti yang telah dijelaskan sebelumnya Google Colab memiliki tampilan yang tidak terlalu berbeda dengan Jupyter notebook. Google Colab dapat menjalankan program untuk setiap cell yang berbeda, cell ini berguna untuk melakukan tracing atau memahami alur dari sebuah program. Dalam modul ini informasi akan dijelaskan pada setiap cell yang ada. Program berikut akan mempraktikkan bagaimana mengevaluasi model yang bertugas mengklasifikasikan citra, apakah citra tersebut citra kucing atau bukan (anjing). Nilai postif adalah artinya kucing dan negatif adalah anjing.

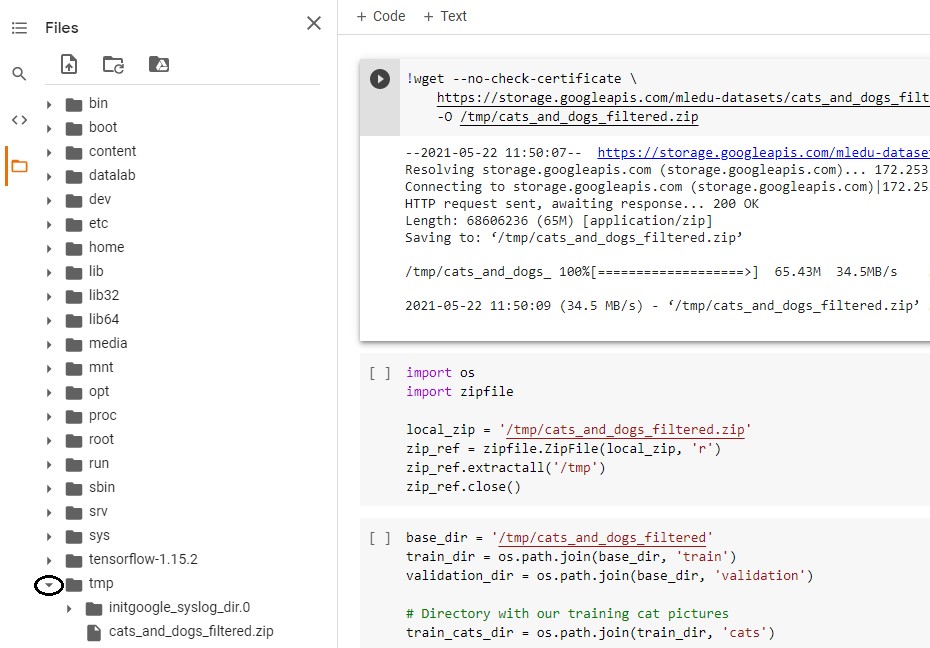
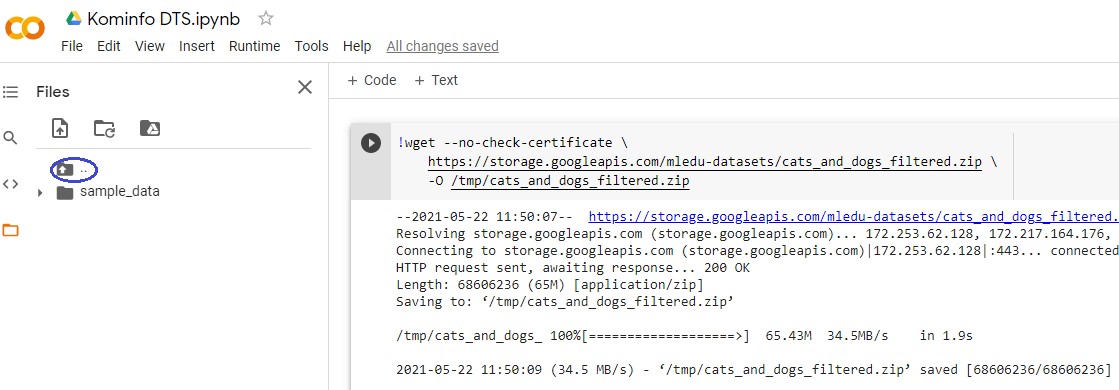
!wget --no-check-certificate \ https://storage.googleapis.com/mledu-datasets/cats\_and\_dogs\_filtered.zip \ -O /tmp/cats\_and\_dogs\_filtered.zip

Source code pada cell ini bertujuan untuk mengambil data foto kucing dan anjing yang akan digunakan sebagai dataset untuk pembuatan model.

|  |
| --- |
| import os import zipfile  local\_zip = '/tmp/cats\_and\_dogs\_filtered.zip' zip\_ref = zipfile.ZipFile(local\_zip, 'r') zip\_ref.extractall('/tmp') zip\_ref.close() |

Setiap data yang didownload pada Google Colab apabila path tidak dideklarasikan maka secara default akan tersimpan di path ‘/tmp/’. Perintah diatas digunakan untuk melakukan unzip pada file cats\_and\_dogs\_filtered.zip yang telah didownload sebelumnya. Cara untuk mengecek foldernya Anda dapat mengikuti cara seperti berikut





|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | base\_dir = '/tmp/cats\_and\_dogs\_filtered' | )  'validation')  'cats') | | train\_dir = os.path.join(base\_dir, 'train' validation\_dir = os.path.join(base\_dir,  train\_cats\_dir = os.path.join(train\_dir, |      |  | | --- | | train\_dogs\_dir = os.path.join(train\_dir, 'dogs') |      |  | | --- | | validation\_cats\_dir = os.path.join(validation\_dir, 'cats') |      |  | | --- | | validation\_dogs\_dir = os.path.join(validation\_dir, 'dogs') | |

Source code berikut untuk mengambil directory dari folder yang nanti-nya akan digunakan.

train\_cat\_fnames = os.listdir(train\_cats\_dir) print(train\_cat\_fnames[:10])

train\_dog\_fnames = os.listdir(train\_dogs\_dir) train\_dog\_fnames.sort()

print(train\_dog\_fnames[:10])

Source code ini digunakan untuk menampilkan 10 nama file yang berada pada folder, hal ini bertujuan untuk memastikan apakah sudah benar data yang kita inginkan sudah didapat oleh program kita.

print('total training cat images:', len(os.listdir(train\_cats\_dir))) print('total training dog images:', len(os.listdir(train\_dogs\_dir))) print('total validation cat images:', len(os.listdir(validation\_cats\_dir))) print('total validation dog images:', len(os.listdir(validation\_dogs\_dir)))

Dengan menggunakan *source code* ini dapat mengetahui berapa banyak foto yang terdapat dalam folder.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | %matplotlib inline  import matplotlib.pyplot as plt import matplotlib.image as mpimg |      |  | | --- | | nrows = 4 ncols = 4 |      |  | | --- | | pic\_index = 0 | |

Source code digunakan untuk menampilakan gambar sebanyak 4 kesamping dan 4 kebawah.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | fig = plt.gcf() fig.set\_size\_inches(ncols \* 4, nrows \* 4)  pic\_index += 8 next\_cat\_pix = [os.path.join(train\_cats\_dir, fname) for fname in train\_cat\_fnames[pic\_index-8:pic\_index]] next\_dog\_pix = [os.path.join(train\_dogs\_dir, fname) for fname in train\_dog\_fnames[pic\_index-8:pic\_index]] |      |  | | --- | | for i, img\_path in enumerate(next\_cat\_pix+next\_dog\_pix):  sp = plt.subplot(nrows, ncols, i + 1) sp.axis('Off') |      |  | | --- | | img = mpimg.imread(img\_path) plt.imshow(img) |      |  |  | | --- | --- | | plt.show() |  | |

Source code berikut digunakan untuk menampilkan 8 foto kucing dan anjing pertama.

from tensorflow.keras import layers from tensorflow.keras import Model

Code berikut digunakan untuk mengambil library yang akan digunakan untuk membuat arsitektur model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | img\_input = layers.Input(shape=(150, 150, 3)) | | x = layers.Flatten()(img\_input) |      |  | | --- | | x = layers.Dense(512, activation='relu')(x) |      |  | | --- | | output = layers.Dense(2, activation='sigmoid')(x) |      |  |  | | --- | --- | | model = Model(img\_input, output) |  | |

Source code berikut adalah code utama untuk pembuatan arsitektur, dalam model ini menggunakan input gambar berwarna dengan dimensi 150x150.

|  |  |
| --- | --- |
| |  | | --- | | model.summary() | |

Source code diatas digunakan untuk mendeklarasikan parameter-parameter yang akan digunakan untuk melakukan proses training.

|  |
| --- |
| from tensorflow.keras.optimizers import RMSprop  model.compile(loss='binary\_crossentropy', optimizer=RMSprop(lr=0.001), metrics=['acc']) |

Source code berikut dijalankan hanya apabila Anda ingin mengecek bagaiman arsitektur dari model yang sudah Anda buat.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | from imutils import paths | '/tmp/cats\_and\_dogs\_filtered/validation')  '/tmp/cats\_and\_dogs\_filtered/train') | | imagePathV = paths.list\_images( imagePathD = paths.list\_images( |      |  |  | | --- | --- | | data = [] | []  [] | | labelsd = val = [] labelsv = | |

Source code berikut digunakan uuntuk mengambil directory folder foto kucing dan anjing, serta mendeklarasikan variabel yang akan digunakan untuk menyimpan gambar dan label.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | from PIL import Image import numpy as np  for imagePath in imagePathD: |      |  | | --- | | a = Image.open(imagePath) image = np.array(a.resize((150,150))) / 255.0 data.append(image) |      |  | | --- | | label = imagePath.split(os.path.sep) [-2] labelsd.append(label) | |

for imagePath in imagePathV:

a = Image.open(imagePath) image = np.array(a.resize((150,150))) / 255.0 val.append(image)

label = imagePath.split(os.path.sep) [-2] labelsv.append(label)

Source code berikut digunakan untuk meresize ukuran dan menormalisasikan nilai warna dari setiap gambar kemudian disimpan kedalam variabel yang telah dibuat sebelumnya. Code juga melakukan penyimpanan label dari setiap foto.

Source code berikut memiliki fungsi yang sama seperti sebelumnya, hanya saja sebelumnya untuk data latih, sedangkan ini untuk data validasi.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | from sklearn.preprocessing import LabelBinarizer e = 55  bprint = 32(labelsv )  lb = LabelBinarizer()  H=model.fit(labelsv1 = lb.fit\_transform(labelsv)trainX, trainY,validation\_data=(testX, testY), batch\_size=b, epochs | |  |
| =e, shuffle=True ) | |
| predY=model.predict(testX)labelsv = np.hstack((labelsv1, 1 - labelsv1))  printlabelsv("\ n Evaluate the new model against the test set:") | |
| model.evaluate(x=testX, y=testY, batch\_size=32) | |
|  | | | |
|  | list\_of\_metricprint(labelsd) | s\_to\_plot = ['accuracy'] |  |
| lb = LabelBinarizer() |
|  | labelsd1 = lb.fit\_transform(labelsd) labelsd = np.hstack((labelsd1, 1 - labelsd1)) labelsd | |  |
|  | |

Source code berikut digunakan untuk merubah bentuk label yang sebelumnya berupa nama kucing atau cat dan anjing atau dog menjadi direpresentasikan dalam bentuk biner.

from sklearn.model\_selection import train\_test\_split

(trainX,testX, trainY, testY) = train\_test\_split(np.array(data), np.array(labelsd), test\_size=0.2) print(trainX.shape) print(testX.shape)

Source code berikut berfungsi untuk memisahkan data secara acak sebagai data latih dan data uji dengan persentase yang telah dapat ditentukan sesuai kebutuhan. Sehingga dengan menggunakan code berikut Anda tidak perlu memisahkan data untuk training dan untuk uji, ketika Anda menyiapkan data hanya perlu memisahkan kelasnya saja.

|  |
| --- |
| e = 55 b = 32    H=model.fit(trainX, trainY,validation\_data=(testX, testY), batch\_size=b, epoch s=e, shuffle=True ) predY=model.predict(testX) print("\n Mengevaluasi Model yang telah ditraining dengan menggunakan data uji  :")  model.evaluate(x=testX, y=testY, batch\_size=32) list\_of\_metrics\_to\_plot = ['accuracy'] |

Source code digunakan untuk melakukan process training dan untuk mengevaluasi model dengan menggunakan data validasi serta menampilkan nilai akurasi yang didapatkan.

|  |
| --- |
| #grafik perbandingan val\_acc dengan train\_acc import matplotlib.pyplot as plt f, ax = plt.subplots() ax.plot([None] + H.history['acc'], 'o-') ax.plot([None] + H.history['val\_acc'], 'x-')  ax.legend(['Train acc', 'Validation acc'], loc = 0) ax.set\_title('Training/Validation acc per Epoch') ax.set\_xlabel('Epoch') ax.set\_ylabel('acc') |
| #grafik perbandingan loss train dengan loss val import matplotlib.pyplot as plt f, ax = plt.subplots() ax.plot([None] + H.history['loss'], 'o-') ax.plot([None] + H.history['val\_loss'], 'x-')  ax.legend(['Train Loss', 'Validation Loss'], loc = 0) ax.set\_title('Training/Validation Loss per Epoch') ax.set\_xlabel('Epoch') ax.set\_ylabel('Loss') |

Kedua source code diatas berfungsi untuk menampilkan grafik dari proses training yang telah dilakukan, dari kedua grafik berikut kita dapat mengetahui apakah model kita sudah melakukan proses training dengan benar. Sehingga dapat mengetahui apakah overfitting ataupun underfitting.

|  |
| --- |
| from sklearn.metrics import multilabel\_confusion\_matrix    predY[1] rounded\_labels=np.argmax(predY, axis=1) rounded\_labels[1] rounded\_predict=np.argmax(testY, axis=1) rounded\_predict[1] |

Source code berfungsi untuk mengolah data validasi dengan data hasil prediksi untuk membentuk Confusion Matrix.

cm = confusion\_matrix(rounded\_labels, rounded\_predict) plot\_confusion\_matrix(cm)

Source code berikut berfungsi untuk membentuk Confusion Matrix, nilai direpresentasikan secara arah jarum jam sebagai berikut: TN, FP, TP, dan FN.

|  |
| --- |
| FP = cm[0,1]  FN = cm[1,0]  TP = cm[1,1]  TN = cm[0,0]    # Sensitivity, hit rate, recall, atau true positive rate  TPR = TP/(TP+FN) #Recall  # Specificity atau true negative rate  TNR = TN/(TN+FP)  # Precision atau positive predictive value  PPV = TP/(TP+FP) #Precision  # Negative predictive value  NPV = TN/(TN+FN)  # Fall out atau false positive rate  FPR = FP/(FP+TN)  # False negative rate  FNR = FN/(TP+FN)  # False discovery rate FDR = FP/(TP+FP)    # Overall accuracy  ACC = (TP+TN)/(TP+FP+FN+TN) F1 = 2\*((PPV\*TPR)/(PPV+TPR)) print("Positi = Kucing, Negatif = Anjing") print(PPV) |

Source code berikut digunakan untuk menghitung dan menampilkan Precision serta Recall.

from sklearn.metrics import classification\_report

print("[INFO] Mengevaluasi model...") predictions = model.predict(testX, batch\_size=32) print(classification\_report(testY.argmax(axis=1), predictions.argmax(axis=1), target\_names=lb.classes\_))

Source code berikut dapat langsung memunculkan nilai dari Precision, Recall, dan F-1 Score.

|  |
| --- |
| from sklearn.metrics import roc\_curve from sklearn.metrics import roc\_auc\_score  # auc scores auc\_score1 = roc\_auc\_score(testY[:,1], predY[:,1])    # roc curve for models  fpr1, tpr1, thresh1 = roc\_curve(testY[:,1], predY[:,1])    # roc curve for tpr = fpr random\_probs = [0 for i in range(len(testY))] p\_fpr, p\_tpr, \_ = roc\_curve(testY[:,1], random\_probs) print('Neural Network : roc auc = %.3f' % (auc\_score1)) |

Source code berikut berfungsi untuk menampilkan nilai dari ROC AUC berdasarkan model yang telah dirancang sebelumnya.

|  |
| --- |
| import matplotlib.pyplot as plt plt.style.use('seaborn')    # plot roc curves plt.plot(fpr1, tpr1, linestyle='--',color='green', label='Neural Network') plt.plot(p\_fpr, p\_tpr, linestyle='--', color='blue')  # title  plt.title('ROC curve')  # x label  plt.xlabel('False Positive Rate')  # y label  plt.ylabel('True Positive rate') |

Program beikur untuk menampilkan grafik dari ROC.

|  |
| --- |
| # precision-recall curve and f1 from sklearn.metrics import precision\_recall\_curve from sklearn.metrics import f1\_score from sklearn.metrics import auc from matplotlib import pyplot  lr\_precision, lr\_recall, \_ = precision\_recall\_curve(testY[:,1], predY[:,1]) lr\_auc = auc(lr\_recall, lr\_precision)  # summarize scores print('Neural Network: auc=%.3f' % (lr\_auc))  # plot the precision-recall curves test\_Y = testY[:,1] no\_skill = len(test\_Y[test\_Y==1]) / len(test\_Y) pyplot.plot([0, 1], [no\_skill, no\_skill], linestyle='--', label='No Skill') pyplot.plot(lr\_recall, lr\_precision, marker='.', label='Neural Network')  # axis labels pyplot.xlabel('Recall') pyplot.ylabel('Precision')  # show the legend pyplot.legend() # show the plot  pyplot.show()    plt.legend(loc='best') plt.savefig('ROC',dpi=300) plt.show(); |

Program berikut digunakan untuk menghasilkan nilai dan menampilkan grafik PR-ROC

from sklearn.metrics import log\_loss

loloss = log\_loss(testY[:,1], predY[:,1])

print('Neural Network : Log Loss = %.3f' % (loloss))

Program berikut untuk menampilkan nilai log loss.

Program dibawah adalah program untuk membuat model regresi sederhana

import numpy as np import pandas as pd import sklearn

import pickle

Import library yang akan digunakan dalam program

|  |  |
| --- | --- |
| |  | | --- | | df = pd.read\_csv('insurance.csv') | |

Untuk import file “insurance.csv” kedalam program

|  |  |
| --- | --- |
| |  | | --- | | df= df[['age', 'sex','smoker','charges']] | |

Melakukan pengambilan data yang akan dijadikan sebagai parameter yang mempengaruhi nilai prediksi

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | df.columns = ['age', 'sex', 'smoker', 'Insurance\_Amount'] |  | |

Mengubah kolom charges menjadi kolom Insurance\_Amount

|  |  |
| --- | --- |
| |  | | --- | | y = df['Insurance\_Amount'] | |

Mengambil nilai dari kolom Insurance\_Amount menjadi nilai target untuk proses training

|  |  |
| --- | --- |
| |  | | --- | | X = pd.get\_dummies(X) | |

Melakukan dummy pada data X, sehingga merubah data dari kolom X yang berisi string akan dipisahkan per-masing-masing stringnya

|  |  |
| --- | --- |
| |  | | --- | | X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33) | |

Program berikut berfungsi untuk memisahkan data train dan test, sebesar 33% untuk data test dan 67% untuk data train.

from sklearn.linear\_model import LinearRegression regressor = LinearRegression() regressor.fit(X\_train,y\_train)

predY=regressor.predict(X\_test)

Proses training data dengan model bernama regressor, kemudian mendapatkan hasil prediksi berdasarkan dari variabel X\_test.

from sklearn.metrics import mean\_absolute\_error print("Mean Absolute Error (MAE):\n") mean\_absolute\_error(y\_test, predY)

Menghasilkan nilai Mean Absolute Error dengan membandingkan variabel y\_test dengan predY.

|  |  |  |
| --- | --- | --- |
|  | | |
|  | airfrom = sklearn.metricspd.read\_csv("AirQuality.csv" import mean\_squared\_error) air.head()import math |  |
|  | MSE = mean\_squared\_error(y\_test, predY) print("Mean Square Error:\n") print(MSE)  RMSE = math.sqrt(MSE) print("Root Mean Square Error:\n") print(RMSE) |  |
|  |

Menghasilkan nilai MSE dan RMSE

Program untuk Clustering

import pandas as pd import numpy as np import matplotlib.pyplot as plt from sklearn.cluster import KMeans from sklearn.preprocessing import MinMaxScaler

Import Library yang akan digunakan

air = pd.read\_csv("AirQuality.csv") air.head()

Melakukan import file “AirQuality.csv” kedalam program untuk melakukan pemrosesan

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | air = air.drop(["Date","Time","CO(GT)","PT08.S1(CO)","C6H6(GT)","PT08.S2(NMHC) | |  |
| ","AH","T","PT08.S5(O3)","PT08.S4(NO2)","NO2(GT)","PT08.S3(NOx)","NOx(GT)","Un |  |
|  | named: 15", "Unnamed: 16"], axis = 1) | |  |
|  | |
| plt.scatter(air.NMHC , air.RH, s =10, c = "c", marker = "o", alpha = 1) air = air.dropna()  plt.show() | |
| air | |
|  | | | |

Melakukan penghapusan untuk beberapa kolom yang tidak akan digunakan dan untuk menghapus nilai NA.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | air.rename(columns={"NMHC(GT)": "NMHC", "RH": "RH"},errors="raise", inplace=Tr | | | ue) |  | |

Melakukan rename pada kolom NMHC(GC) menjadi NMHC

air.info()

Source code berikut digunakan untuk mengecek tipe data dan banyaknya data.

plt.scatter(air.NMHC , air.RH, s =10, c = "c", marker = "o", alpha = 1) plt.show()

Untuk menampilkan grafik scatter plot dari data yang digunakan

air\_x = air.iloc[:, 0:2] air\_x.head()

Melukan pengubahan bentuk data dan melakukan pengecek data teratas.

x\_array = np.array(air\_x) print(x\_array)

Untuk mengubah variabel dari dataframe ke numpy array, kemudian menampilkan data array

scaler = MinMaxScaler()

x\_scaled = scaler.fit\_transform(x\_array) x\_scaled

Menormalisasi data dengan metode MinMax

kmeans = KMeans(n\_clusters = 5, random\_state=14) kmeans.fit(x\_scaled) print(kmeans.cluster\_centers\_) label=kmeans.predict(x\_scaled)

Melakukan training untuk model clustering dengan menggunakan metode KMeans dengan nilai n sama dengan 5 dan random state 14

air["kluster"] = kmeans.labels\_ output = plt.scatter(x\_scaled[:,0], x\_scaled[:,1], s = 100, c = air.kluster, m arker = "o", alpha = 1, ) centers = kmeans.cluster\_centers\_ plt.scatter(centers[:,0], centers[:,1], c='red', s=200, alpha=1 , marker="o"); plt.title("Hasil Klustering K-Means") plt.colorbar (output) plt.show()

Menampilkan plot hasil dari model clustering.

from sklearn.metrics import silhouette\_score

print(f'Silhouette Score(n=5): {silhouette\_score(x\_scaled, label)}')

Menampilkan hasil dari nilai evaluasi model dengan metode Silhoutte Score