**PENGENALAN PEMBELAJARAN MESIN**

**UNTUK ANALISA DATA**

1. **Apa itu pembelajaran ?**

Istilah pembelajaran muncul pada algoritma dewasa ini. Contoh sederhana berikut memberikan motivasi bagaimana algoritma dibangun. Misalkan kawanan tikus mencari makanan dan menemukan secara tiba-tiap dengan kelihatan bagus dan menarik, maka tikus akan memakannya sedikit demi sedikit. Jika makanan terasa enak, maka tikus akan melanjutkan makan. Jika tikus merasakan makanan tidak enak, maka makanan tidak dimakan. Jika makanan memuat racun dimakan tikus, maka pada periode tertentu tikus tidak mencari makanan pada lokasi dimana makanan tikus diperoleh. Ada proses pembelajaran secara natural dimana tikus dapat mengenali bahwa makanan pada lokasi yang sama beracun sehingga tidak datang pada lokasi tersebut untuk periode tertentu.

Diinspirasi oleh sifat natural tersebut, misalkan kita dapat membangun algoritma untuk mengkoleksi email yang tergolong spam. Mesin akan mengingat bagaimana berdasarkan pengetahuan sebelumnya bahwa email dikatakan spam oleh pengguna maka email yang datang selanjutnya dapat tergolong spam dan bukan spam. Jadi pendekatan cara kerja demikian dikatakan ‘pembelajaran karena pengingat’. Hal ini mempunyai kekurangan dalam aspek pembelajaran yaitu kemampuan memberi label pada pesan email yang tdak terlihat. Suatu pembelajaran berhasil jika dapat melakukan kemajuan secara individu dalam melakukan perumuman yang lebih luas. Untuk mencapai perumuman dalam tugas melakukan filterisasi email pada spam, pembelajar dapat menelusur email-email yang sebelumnya dilihat dan melakukan ekstraksi kata-kata dalam pesan yang terindikasi spam. Ketika email baru datang, mesin dapat menguji apakah kata-kata dalam email tersebut sbagai spam dan menduga labelnya. Sistem yang demikian dapat menduga secara benar pelabelan dalam email-email yang tidak terlihat. Cara seperti ini disebut cara berpikir induktif. Akan tetapi cara induktif ini bisa selalu benar sebagaimana dijelaskan pada kasus perilaku merpati berikut.

Kasus ini dikenal sebagai *Pigeon Superstition*. Seorang tokoh psikolog pada bidang perilaku dari Amerika (Skinner,tahun) menempatkan sekumpulan burung merpati dalam suatu sangkar. Mekanisme otomatis dilakukan yaitu mengirimkan makanan dalam sangkar pada interval waktu tertentu tanpa memperhatikan perilaku burung-burung tersebut sebelumnya. Burung-burung tersebut makan pada makanan yang pertama kali dikirimkan dan setiap burung melakukan beberapa aktivitas (mengambil makanan, memutar kepala, dst). Setiap awal pengiriman makanan, srtiap burung melakukan aksi spesifik dan akibatnya setiap burung cenderung memerlukan waktu untuk melakukan aksi yang sama. Kemudian, perubahan pengiriman makanan dilakukan secara random, setiap burung melakukan hal yang sama, pada saat awal bagaimana makanan dikirimkan. Hal ini dilakukan dengan disiplin oleh burung-burung itu. Video yang terkait dengan penjelasan Sninner dijelaskan pada youtube (Web2). Pada video ditunjukkan bagaimana burung dalam eksperimen diberi makan pada awal dengan suara pemberian makan tertentu. Jika pemberian makan diberi kode dengan cahaya, maka burung tidak menunjukkan perilaku mengambil makanannya. Pada video serupa, seorang anak diberi makanan dari suatu meja. Jika meja ditinggikan, anak tersebut secara otomatis mengambil kursinya untuk dapat meraih makanan yang semula terjangkau tetapi menjadi tidak terjangkau.

Mekanisme pembelajaran apa yang membedakan antara perilaku burung dengan pembelajar pada mannusia ? Hal ini merupakan pertanyaan krusial dalam pengembangan pembelajar otomatis. Pembelajar manusia dapat bersandar pada *common sense*(pemahaman umum) untuk melakukan filter terhadap pembelajaran yang tidak bermakna secara random. Jika kita sudah menyerahkan tugas demikian pada mesin, kita harus membuat prinsip crisp (pengkodean) yang baik yang akan melindungi program dari pencapaian kesimpulan yang tidak bermakna. Pengembangan prinsip demikian merupakan pusat teori dari mesin pembelajaran (*machine learning*). Yang menjadi masalah : mengapa pada tikus berhasil menggunakan induksi sedangkan pada burung merpati tidak bisa berlaku ? Hal inilah yang dipelajari dalam ML. Perbedaan keduanya inilah yang membuat bias dalam mekanisme pembelajaran dan juga bias dalam induksi.

Pembelajaran mesin digolongkan sebagai aplikasi kecerdasan buatan yang menyediakan sistem mampu secara otomatis mempelajari dan memperbaiki dari pengalaman tanpa secara eksplisti deprogram. Pembelajaran mesin focus pada perkembangan komputer program yang dapat mengakses data dan menggunakannya untuk dirinya sendiri. Pross pembelajaran dimulai dari mengobservasi data dari langsung pengamaan atau berdasarkan instruksi untuk melihat pola data dan membuat keputusan yang lebih baik untuk mendatang berdasarkan contoh-contoh yang disediakan. Tujuan utama adalah untuk mengijinkan komputer mempelajari secara otomatis tanpa intervensi manusia dan dapat melakukan aksi penyesuaian yang terkait.

* 1. **Kapan memerlukan Mesin Pembelajaran (*Machine Learning*)**

Kapankah ML perlu digunakan ? Terdapat beberapa masalah yang memerlukan ML yaitu :

1. Tugas terlalu kompleks untuk dipogram

Dengan perkembangan sekarang ini masalah yang timbul karena perilaku hewan/manusia sangat besar sehingga tidak mudah untuk diselesaikan

1. Tugas melebihi kapasitas manusia

Tugas pengenalan bicara, pengenalan gambar, data astronomi, pengenalan cuaca, informasi medis, data analisis gen, data web, *e-commerce*, merupakan contoh dimana manusia tidak bisa melakukan secara langsung.

1. Adaptif

Biasanya, program dibuat dan diinstal maka program kemudian tidak berubah Lain halnya program dalam ML dimana program dapat diubah-ubah. Penyimpanan data yang besar akan dapat dikenali polanya dan kombinasi data memerlukan proses komputer yang cepat dan terbuka. Selain itu, banyak tugas yang berubah dari waktu ke waktu dari pengguna data yang satu ke pengguna yang lain karena ada interaksi antar pengguna.

Oleh karena kompleksitas dalam ML, maka pembelajaran dibedakan dalam beberapa tipe yaitu Pembelajaran terbimbing (*supervised learning*) dan Pembelajaran tak terbimbing (*unsupervised learning*) dimana keduanya akan dijelaskan pada subbab 2

.

* 1. **Kompleksitas dengan bidang lain**

Karena kompleksitas pemanfaatannya maka ML terkait dengan beberapa bidang yaitu statistik, teori informasi, teori permainan dan optimasi. Sebagai bagian dari komputer sains, ML dapat dipandang sebagai bagian dari teori kecerdasan buatan (*artificial Intelegence*). dimana ML dapat membuat pengalaman menjadi suatu keahlian atau mendeteksi pola yang bermakna dalam sensor data yang kompleks dari intelegensi manusia dan hewan.

1. **Pengenalan Algoritma Mesin Pembelajaran** (*Machine Learning*)

Istilah mesin pembelajaran dari bahasa Inggris *Machine Learning* dimana istilah ini muncul dalam studi analisa data yang dikatakan big data. Demikian pula terdapat beberapa istilah yang akan muncul yang mengkategorikan data sebagai berikut

**2.1 Mesin Pembelajaran Terbimbing** (SML : *Supervised Machine Learning*)

Ada 2 istilah dasar dalam diskusi berikut yaitu *Supervised Learning* (SL) dan *Unsupervised Learning* (UL) dimana dalam ML sering menggunakan SL. Marilah kita pahami masing-masing istilah tersebut :

SL adalah algoritma yang mempunyai variabel input *(x*) dan variabel output/target (*Y*) dan kita menggunakan algoritma untuk melakukan pemetaan fungsi dari input ke-output yaitu

*Y* = *f*(*x*)

Tujuannya adalah mendekati fungsi pemetaan sebaik mungkin sehingga input baru *x* dapat memprediksi luaran *Y* untuk data tersebut. Tentunya mahasiswa pernah melakuan hal ini. Proses algoritma ini dikatakan SL karena proses algoritma menggunakan himpunan data latih (data training) yang dianggap sebagai pengajar yang membimbing dalam proses belajar. Algoritma secara iteratif membuat prediksi pada data latih dan dikoreksi oleh pengajar. Proses pembelajaran berhenti ketika algoritma pada level yang dapat diterima.

Catatan : perhatikan bahwa istilah pengajar disini bukan personal selaku seseorang tetapi merupakan bagian dari algoritma.

Masalah SL dinyatakan dalam 2 grup yaitu masalah regresi dan masalah klasifikasi.

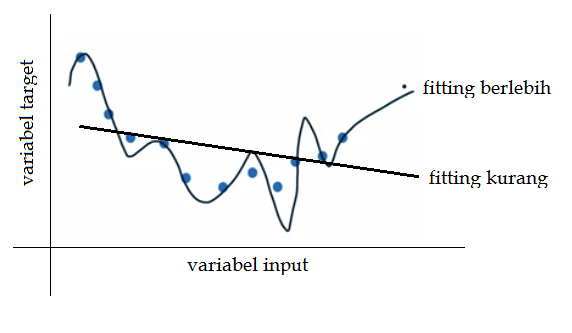
Masalah klasifikasi : ketika variabel output sebagai variabel kategori seperti merah atau biru, berpenyakit atau tidak berpenyakit.

Masalah regresi : ketika variabel outpur berupa nilai real seperti nilai rupiah, berat.

Beberapa algoritma SML adalah :

1. regresi linear untuk masalah regresi
2. Random forest untuk masalah klasifikasi dan regresi
3. SVM (Support Vector Machines) untuk masalah klasifikasi

Dalam literatur yang lain kita dapat menjumpai bahwa ketiganya lebih dinyatakan secara detail (Web 3). Sebenarnya secara keilmuan, beberapa materi sudah pernah dipelajari seperti regresi yang dinyatakan dalam regresi linear dan nonlinear (polynomial regresi). Sedangkan nama variabel input juga variabel input dan variabel output dikatakan variabel target. Demikian dalam regresi digunakan istilah *underfitting* dan *overfitting* sebagaiamana pada Gambar 1.



**Gambar 1.** Illustrasi fitting kurang dan fitting berlebih

Daftar SL yang dipelajari secara praktis dalam internet (web 3) dinyatakan dalam Tabel 1 dimana daftar dalam bahasa Inggris dan dalam bahasa Indonesia.

**Tabel 1** Daftar SL daftar dalam bahasa Inggris dan dalam bahasa Indonesia.

|  |  |
| --- | --- |
| Daftar dalam bahasa Inggris | Daftar dalam bahasa Indonesia |
| *Overfitting and Underfitting*  *Supervised Learning: Datasets*  *K-Nearest Neighbors: Classification and Regression*  *Linear Regression: Least-Squares*  *Linear Regression: Ridge, Lasso, and Polynomial Regression*  *Logistic Regression*  *Linear Classifiers: Support Vector Machines*  *Multi-Class Classification*  *Kernelized Support Vector Machines*  *Cross-Validation*  *Decision Trees* | Fitting berlebih dan fitting yang kurang  SL dengan data  Persekitaran terdekat sebesar *K* : klasifikasi dan Regresi :  Regrei linear : Kuadrat terkecil  Regrssi linear : Ridge , Lasso dan polynomial  Regresi Logistik  SVM  MCC  SVM Kernel  Validasi bersilang  Pohon Keputusan |

Sebenarnya SL untuk tingkat lanjut, masih memuat beberapa bagian sebagaimana pada Tabel 2 yang ditulis dalam bahasa Inggris agar pembaca dapat merujuk dengan kata kunci yang sama dimana memuat algoritma pohon keputusan.

**Tabel 2** Daftar SL daftar dalam bahasa Inggris dan dalam bahasa Indonesia tingkat lanjut

|  |  |
| --- | --- |
| Daftar SL tingkat lanjut dalam bahasa Inggris | Daftar SL tingkat lanjut dalam bahasa Indonesia |
| [*Naive Bayes Classifiers*](https://www.coursera.org/lecture/python-machine-learning/naive-bayes-classifiers-0XFms)  *Random Forests*  *Gradient Boosted Decision Trees*  *Neural Networks*  *Deep Learning (Optional)*  *Data Leakage*  *Dimensionality Reduction and Manifold Learning*  *Clustering* | Klasifikasi Bayes standard  Hutan acak  Pohon keputusan yang ditingkatkan kemiringannya  Pembelajaran dalam  Kebocoran data  Pengurangan dimensi dan berlapis  Klastering |

**2.2 Pembelajaran Mesin Takterbimbing** (*Unsupervised Machine Learning* (UML))

Pada UML kita hanya punya data input tetapi tidak punya output. Tujuan UML adalah suatu model berdasarkan struktur atau distribusi data untuk mempelajari lebih lanjut tentang data tersebut. Metode ini disebut UML karena tidak ada jawaban yang benar dan tidak ada pengajar yang mengkoreksi. Algoritma menyerahkan pada alat yang digunakan dalam menemukan sesuatu dan menghasilkan struktur yang menarik dari data.

UML dapat digolingkan dalam masalah klastering dan asosiasi.

1. Masalah klastering: Masalah klatering merupakan upaya menemukan pengelompokan dalam data sehingga diperoleh kelompok yang sifatnya dikehendaki
2. Aturan asosiasi : masalah aturan asosiasi adalah ketika kita menghendakii menemukan aturan yang menjelaskan porsi besar data. Misal suatu kelompok orang yang menghendaki membeli barang X tetapi juga membeli barang Y.

Algoritma yang terkait dengan UML diantaranya :

1. k-means clustering
2. algoritma apriori untuk aturan asosiasi

**2.3 Semi Mesin Pembelajaran Terbimbing (*Semi-Supervised Machine Learning* (SSML))**

Masalah dengan input data yang sangat besar dan hanya beberapa data berlabel tergolong pada masalah SSML. Misalkan suatu foto hanya sebagian yang diberi label sedang sebagian besar tidak. Masalah nyata seringkali tergolong pada masalah sepertini ini. Hal ini muncul umumnya karena waktu yang besar untuk melabel data dan memerlukan akses pada domain data, sedangkan data yang tidak berlabel murah dan mudah untuk dikoleksi dan disimpan.

Kita dapat menggunakan USL untuk menemukan struktur data input kemudian kita menggunakan teknik SL untuk menduga data takberlabel. Data tersebut kemudian kita gunakan dalam algoritma SL sebagai data training dan menggunkaan model untuk membuat prediksi pada data baru yang tidak terlihat.

**Kesimpulan Sementara**

Sebagai kesimpulan untuk diskusi di atas, kita dapat mengatakan :

SL : Semua data berlabel dan algoritma belajar untuk memprediksi luaran dari data input

UL: semua data tidak berlabel dan algoritma mempelajari struktur data input.

SSL : beberapa data berlabel tetapi sebagian besar tidak berlabel dan campuran metode SL dan USL digunakan.

Apakah selama ini para peneliti belum melakukan pengembangan hal ini sebelum istilah atau ilmu ML dikembangkan ? jika dicermati ML muncul sehubungan kompleksitas data yang timbul dan perubahan yang berlaku secara simultan dengan perubahan data serta sangat cepat. ML memberikan generalisasi terhadap beberapa metode klasik yang sudah ada seperti regresi dan klastering serta memberikan pengembangan terhadap metode klasik tersebut dengan bahasa pemrograman yang *up to date* saat ini.

1. **Langkah-langkah Pengolahan data dalam bahasa mesin**

Berikut ini diberikan langkah-langkah praktis dalam pengolahan data dengan bahasa mesin

**3.1 Pembacaan Data**

Tahap pembacaan data misalkan Menampilkan data, menampilkan dimensi data, menghitung tiap klas, Menampilkan data per faktor/variabel.

* 1. **Membersihkan data**

Membersihkan data diperlukan dalam pengolahan data machine learning dimana data dapat dipertanggungjawabkan untuk kemudian dapat diolah. Beberapa proses membersihkan data dikarenakan :

1. Observasi ganda ataupun tidak relevan.
2. Salah dalam pelabelan , kategori yang sama terjadi berulang.
3. Data hilang atau ada data yang kosong (null data)
4. Terdapat outlier yang tidak diharapkan.

Pada kegiatan ini dapat dianggap bahwa tentang bagian a atau b sudah baik (observasi relevan dan tidak salah dalam pelabelan). Untuk itu kita mengobservasi ada tidaknya data yang hilang.

**3.2.1 Mencermati data outlier**

Data –data diatas dibaca dan dikatakan invalid. Untuk mengatasi data invalid, maka diperlukan langkah-langkah sebagai berikut :

1. Abaikan/hapus data invalid : Hal ini jelas tidak mungkin pada hampir semua kasus karena hal ini berarti kehilangan informasi yang berharga.
2. Membuat rata-rata : pada data yang invalid diisi data rata-rata . Hal ini kemungkinan dapat berlaku pada beberapa data, tetapi dengan menggantikan nilai rata-rata dapat memberikan informasi yang salah pada model.
3. Menghindari menggunakan faktor-faktor tersebut : hal ini mungkin dapat dilakukan jika data banyak memuat informasi yang invalid. Misalkan hal ini berlaku pada ketebalan kulit tetapi kemudian kita juga sulit melakukan prediksi.
   * 1. **Mencermati data yang hilang**

Sudah jelas dari program dapat ditunjukkan hasil data yang hilang

**3.3 Proses Teknis lebih lanjut**

Proses teknis lebih lanjut adalah proses mentransformasi data menjadi lebih baik dalam mempresentasikan masalah yang akan diselesaikan dalam model sehingga memperbaiki model dan akurasi model.Proses ini memberikan lebih faktor-faktor sebagai input terhadap faktor-faktor yang sudah ada dan menggambungkan beberapa faktor agar menghasilkan faktor-faktor yang secara intuitif (karena belum dibuktikan) pada model. Selain itu penambahan faktor diharapkan tidak memberikan *overfitting*. Karena data sudah disediakan dan dengan mencermati data seperti di atas, kita tidak dapat membuat atau menghilangkan data pada langkah ini.

Umumnya faktor teknis ditunjukkan sebelum pemilihan model. Akan tetapi pada tulisan ini kita melakukan pendekatan yang berbeda. Faktor teknis (ditulis dalam bahasa Inggris *adalah Feature Enginerring* ) .Pendekatan tersebut yaitu mencermati semua faktor dalam data dan mendiskusikan seberapa penting faktor-faktor tersebut dalam model.

3.4 **Pemilihan Model**

### Pemilihan model atau algoritma pemilihan merupakan algoritma yang paling menarik dan menjadi jantung utama dalam *machine learning*. Hal ini merupakan fase dimana kita memilih model dan menampilkan data secara terbaik. Langkah awal yang dilakukan adalah melakukan Uji Akurasi dari model-model klasifikasi berdasarkan parameter yang sudah ditetapkan untuk tiap model dikatakan baik.

Algoritma utama untuk hal ini adalah  **K-Nearest Neighbors, Support Vector Classifier, Logistic Regression, Gaussian Naive Bayes, Random Forest and Gradient Boost**  sebagai pengklasifikasi model terbaik.

* + 1. **Mengevaluasi Model**

Pada langkah praktis yang umum, maka kita perlu menghindari data latih dan data uji pada data yang sama. Model mempunyai tujuan untuk memprediksi data yang diluar sampel data dan model dapat kompleks sehingga dapat terjadi *overfitting*. Untuk menghindari hal-hal demikian, terdapat 2 langkah awal :

1. Memisahkan data menjadi data latih dan data uji yang disebut *train/test method*
2. Melakukan validasi silang *K*-kali yang disebut *K-Fold Cross Validation*

Selanjutnya algoritma akan memberikan nilai akurasi masing-masing model dari langkah ke-2 di atas untuk mengevaluasi akurasi dari model dalam metode train/test

Mengimport fungsi-fungsi yang diperlukan dengan python adalah :

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

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import KFold # ini ditambahkan krn tidk ada tetapi perlu

Selanjutnya kita memilih nilai akurasi tertinggi yang menunjukkan model terbaik.

1. **Pemisahan data latih dan data uji**

Pada langkah ini kita akan membuat data menjadi 2 yaitu data latih (training set) dan data uji (testing set) dimana data uji untuk menguji model dan mengevaluasi akurasi.

**Pemisahan dengan Scikit Learn**

Jadi kita akan memisahkan faktor-faktor dan respon dalam data latih dan data uji. Kita melakukan proses ini dimana setiap klas respon (misal yang berlabel diabetes) diwakili dengan porsi yang sama pada tiap sampel data.

Selanjutnya kita akan memasangkan data terhadap setiap model dan menghitung akurasi setiap model dengan nilai akurasi.

1. **Validasi silang (*K*- cross validation)**

Metode ini melakukan pemisahan data menjadi partisi sebanyak *K*-partisi yang sama dan menggunakan 1 lapisan (*fold*) sebagai data uji dan kemudian menggabungkan partisi yang lain sebagai data latih. Kemudian model diuji akurasinya. Proses ini akan dilakukan K-kali dengan menggunakan partisi yang berbeda sebagai data uji setiap waktu. Akurasi rata-rata uji dalam proses merupakan akurasi hasil uji model. Dengan ini estimasi untuk data diluar data sampel akan lebih akurat serta dikatakan lebih efisien dalam menggunakan data (setiap observasi digunakan untuk data latih dan data uji). Akan tetapi kendala yang dihadapi bahwa metode ini lebih lambat dibandingkan metode pemisahan di atas (Train /Test Split)

Proses ini digambarkan pada Gambar 2 .

Data

Data uji (test) (Te)

Data latih (*training*) (Ta)

(Te)

(Te)

Gambar 2. Partisi data dengan *K*=3 kali partisi

Perhatikan bahwa dalam penulisan di atas kita hanya belajar dalam membaca data, membersihkan data dan mengevaluasi data terhadap beberapa model sehingga dapat dipilih secara otomatis atas model-model yang dipilih. Terdapat banyak model yang sudah diujicobakan terhadap data tersebut dimana kita belum menelusur masing-masing model secara lebih detail, yaitu **K-Nearest Neighbors, Support Vector Classifier, Logistic Regression, Gaussian Naive Bayes, Random Forest and Gradient Boost** akan tetapi telah dipelajari melalui video. Pemilihan data terhadap model Logistic Regression menunjukkan bahwa model yang lain dapat *underfitting* atau *overfitting*. Selain memahami masing-masing model lebih detail, maka hasil pemodelan terhadap data dengan LR perlu diperhatikan lebih lanjut.

**3.5 Proses Pemodelan data yang sudah bersih**

Sebagaimana disebutkan bahwa data telah dibersihkan kita dapat langsung memproses pengolahan data pada program yang baru dengan hasil identifikasi digunakan untuk langsung kembali memanggil data

1. **Studi Kasus Dan Teknik Pemrograman Python Dengan Algoritma Mesin Pembelajaran**

**4.1 *Overfitting* dan *Underfitting***

Secara sederhana *overfitting* (OF) dan *underfitting (UF)* merupakan proses memodelkan data untuk menyesuaikan dengan kurva tertentu yang diasumsikan dapat menggambarkan data. Kata over menyatakan sifat yang kelebihan dimana kurva yang digunakan mempunyai derajat ketelitian yang terlalu banyak. Jika pola data linear, tidaklah tepat mengasumsikan data berpola kuadratik atau polynomial derajat lebih tinggi. Kasus demikian dikatakan *overfitting*. Demikian pula sebaliknya, jika data berpola tidak linear (sinusoidal) memuat *noise*, tidak mungkin menggunakan model linear. Kasus demikian yang disebut underfitting.

*Overfitting* dan *underfitting* dapat terjadi juga dalam kegiatan klasifikasi seperti *clustering* dimana salah satu metode menggunakan kurva sebagai batas pembeda antar klaster. Demikian pula setelah melakukan *fitting* tentu diperlukan evaluasi terhadap model yang diperoleh dimana pada *machine learning* digunakan metode *cross validation* yang secara mudah diterjemahkan sebagai validasi silang. Dari istilah ini kita diinspirasi bahwa model yang ada kemudian divalidasi dengan mencocokan terhadap data.

Program yang memberikan contoh *overfitting* (OF) dan *underfitting* (UF) diberikan pada internet (Web 4).

Yang ditunjukkan berikut ini.

Misalkan kita membangkitkan data sebagai fungsi + dimana  dan sebagai bilangan random agar data yang dibangkitkan dari fungsi memuat gangguan (*noise*). Kemudian diasumsikan bahwa data(pasangan (*x,y*)) tersebut sebagai fungsi polinomial derajat 1, 4 dan 15. Tentu saja polinomial derajat 1. Secara langsung, telah diketahui data tidak berpola linear sehingga pendekatan sebagai fungsi polinomial derajat 1 masih kurang tepat. Pendekatan ini dikatakan underfitting. Sedangkan polinomial derajat 4 dan 15 merupakan kandidat yang dapat dianggap tepat menyatakan fungsi cosinus di atas. Akan tetapi jika polynomial derajat 4 sudah terlihat cukup baik secara visual menyatakan fungsi cosinus di atas, maka polynomial derajat 15 mendekati data tersebut terlalu berlebihan. Perhatikan bahwa menyatakan fungsi sebagai polynomial derajat 15 berarti terdapat 16 parameter polinomial yang harus ditentukan. Secara komputasi, polinomial derajat 15 lebih kompleks dibandingkan menentukan polynomial derajat 4.

Program =1

print(\_\_doc\_\_)

import numpy as np

import matplotlib.pyplot as plt

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import cross\_val\_score

def true\_fun(X):

return np.cos(1.5 \* np.pi \* X)

np.random.seed(0)

n\_samples = 30

degrees = [1, 4, 15]

X = np.sort(np.random.rand(n\_samples))

y = true\_fun(X) + np.random.randn(n\_samples) \* 0.1

plt.figure(figsize=(14, 5))

for i in range(len(degrees)):

ax = plt.subplot(1, len(degrees), i + 1)

plt.setp(ax, xticks=(), yticks=())

polynomial\_features = PolynomialFeatures(degree=degrees[i],

include\_bias=False)

linear\_regression = LinearRegression()

pipeline = Pipeline([("polynomial\_features", polynomial\_features),

("linear\_regression", linear\_regression)])

pipeline.fit(X[:, np.newaxis], y)

# Mengevaluasi model dengan cross validasi

scores = cross\_val\_score(pipeline, X[:, np.newaxis], y,

scoring="neg\_mean\_squared\_error", cv=10)

X\_test = np.linspace(0, 1, 100)

plt.plot(X\_test, pipeline.predict(X\_test[:, np.newaxis]), label="Model")

plt.plot(X\_test, true\_fun(X\_test), label="Fungsi yang benar")

plt.scatter(X, y, edgecolor='b', s=20, label="Data sampel")

plt.xlabel("x")

plt.ylabel("y")

plt.xlim((0, 1))

plt.ylim((-2, 2))

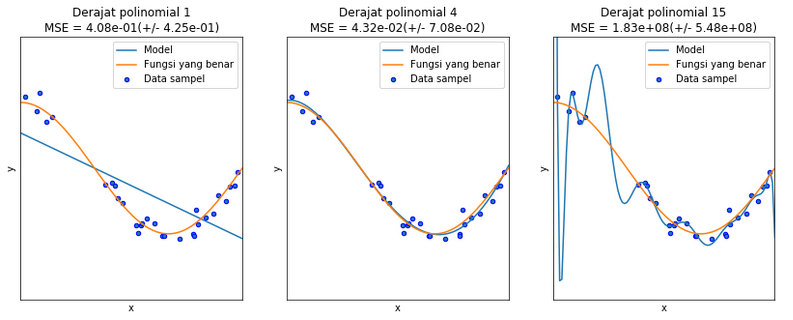
plt.legend(loc="best")

plt.title("Derajat polinomial {}\nMSE = {:.2e}(+/- {:.2e})".format(

degrees[i], -scores.mean(), scores.std()))

plt.show()

Hasil luaran ditunjukkan berikut ini



Gambar 3. Ilustrasi *underfitting* (kiri), *overfitting* (kanan) dan *fitting* yang tepat (tengah)

Pada Gambar 3 ditunjukkan bagaimana kedua istilah yaitu *underfitting* dan *overfitting* diillustrasikan. Pola data dapat diilustrasikan secara masuk akal dengan polinomial derajat 4.

Sebelum kita melakukan penjelasan tahap-tahap pengolahan data dalam bahasa mesin maka berikut ini perlu dijelaskan bagaimana proses pengolahan data dengan bahasa mesin dilakukan.

* 1. **Pengolahan data dengan algoritma mesin pembelajaran**

Pada bagian ini akan ditunjukkan taahapan yang dijelaskan diatas untuk data observasi yang diperoleh di internet yaitu untuk observasi terhadap penyakit diabetes.

* 1. **1 Latar Belakang**

Diabetes dikenal sebagai penyebab kematian dini dan disabilitas. Oleh karena itu, diabetes merupakan salah satu dari 4 penyakit yang ditargetkan untuk dicegah dan dikontrol dalam dunia sejak 2013 menurut WHO. Pada tahun 2012 tercatat bahwa 1.5 kematian manusia di bumi diikarenakan diabetes dimana sebagai tingkat ke-8 penyebab kematian dan sebagai penyebab kematian pada wanita tingkat ke-5 di dunia.

Diabetes merupakan salah satu penyakit yang dianggap kronis ketika pancreas tidak berfungsi dalam menghasilkan insulin (hormon yang mengatur gula darah) dan atau ketika tubuh tidak efektif memproduksi insulin. Ada 2 tipe diabetes yang juga dikenal. Tipe 1 diabetes dikenal sebagai kebergantungan insulin yang dicirikan oleh penurunan produksi insulin sehingga penderita harus secara teratur mengkonsumsi insulin untuk mengatur glukosa dalam darah. Tipe ini tidak diketahui penyebabnya dan masih belum dapat dicegah. Untuk penderita diabetes tipe 2 disebut juga penderita yang bergantung pada non-insulin yang sebagian besar terjadi pada manusia dewasa dimana gejalanya seperti pada tipe 1 hanya saja tidak terdiagnosa pada beberapa tahun hingga terjadi komplikasi. Jenis lain adalah bentuk transisi antara tipe 1 dn tipe 2 yang disebut *Impaired glucose tolerance* (IGT) dan *impaired fasting glycaemia* (IFG) dimana orang dalam kondisi ini seringkali dalam kondisi gula darah tidak normal sehingga terjadi serangan jantung dan stroke. Demikian pula *Gestational* diabetes (GDM) merupakan salah satu jenis penyakit diabetes dengan kondisi temporer yang mana terjadi pada kehamilan. Hal ini terjadi ketika glukosa dalam darah bernilai di atas normal tetapi masih kondisi dibawah terdiagnose sebagai penderita diabetes. Resiko meningkat selama kehamilan dan dapat diturunkan kepada calon bayi.

Dengan memperhatikan tingginya pengaruh penyakit diabetes terhadap kematian manusia di dunia, penelitian terhadap pencegahan dan penyembuhan terhadap penyakit ini telah banyak dilakukan oleh para peneliti. Salah satu data penelitian telah dinyatakan di internet terhadap 768 orang wanita dengan 9 faktor dari *National Institute of Diabetes and Digestive and Kidney Diseases*. Data dicatat baik terhadap pasien yang terdiagnosa diabetes ataupun tidak. Beberapa faktor tersebut akan ditinjau lebih detail dalam tulisan ini. Faktor yang diukur tentunya faktor-faktor yang dianggap berpengaruh secara signifikan terindikasi tidaknya seseorang sebagai penderita diabetes. Oleh karena itu keadaan hamil atau tidak dicatat sebagai salah satu faktor. Adapun faktor-faktor selengkapnya adalah sebagai berikut :

Kehamilan : Banyaknya kehamilan yang pernah dialami;

Glukosa : Konsentrasi glukosa dalam 2 jam tes toleransi glukosa dalam uji gula darah

Tekanandarah : tekanan darah dalam keadaan diastolik (mm/Hg)

Tebal kulit : ketebalan lipatan dalam otot dibawah kulit (mm)

Insulin : serapan serum insulin dalam 2 jam (mu U/ml satuan)

BMI : (Body mass index) = berat dalam kg/tinggi dalam meter kuadrat)

Fungsi pedigree : merupakan sejarah diabetes mellitus yang relatif terkait dengan faktor genetik

Umur : umur pasien (dalam tahun)

Luaran : variabel klas (0 atau 1), 1 : untuk pasien terdiagnose diabetes, 0 : untuk pasien terdiagnose tidak diabetes

BMI pada umumnya pada 18.5-24.9 dikatakan normal sedangkan dalam nilai 25-29.9 dikatakan berat berlebih dan jika > 30 maka dikatagorikan obesitas.

* 1. **2 Langkah-langkah Pengolahan data**

**4.2.2.1 Pembacaan Data**

Penamaan variabel untuk 9 faktor diatas sebagai berikut : Kehamilan, Glukosa , Tekanan darah Tebal kulit, Insulin ,BMI, Fungsi Pedigree, Umur, Luaran.

Contoh daftar data ditunjukkan pada Tabel 1.

**Tabel 1.** Contoh data yang dianalisa hasil observasi oleh *National Institute of Diabetes and Digestive and Kidney Diseases* terhadap 768 wanita

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Kehamilan | Glukosa | Tekanandarah | Tebalkulit | Insulin | BMI | FungsiPedigree | Umur | Luaran |
| 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |
| 5 | 116 | 74 | 0 | 0 | 25.6 | 0.201 | 30 | 0 |
| 3 | 78 | 50 | 32 | 88 | 31 | 0.248 | 26 | 1 |
| 10 | 115 | 0 | 0 | 0 | 35.3 | 0.134 | 29 | 0 |

Langkah-langkah pemrograman sebagai berikut

**Langkah 1.** Program Python Memanggil fungsi

https://towardsdatascience.com/machine-learning-workflow-on-diabetes-data-part-01-573864fcc6b8

%matplotlib inline

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

diabetes= pd.read\_csv('E:/JAN2016/KULIAH20182019/SEM2\_20182019/ANALISA\_DATA/PROJECTAnalisaData/diabetesIndo.csv')

diabetes.columns

Hasil program :

Index([u'Kehamilan', u'Glukosa', u'Tekanandarah', u'Tebalkulit', u'Insulin',

u'BMI', u'FungsiPedigree', u'Umur', u'Luaran'],

dtype='object')

**Langkah 2.** Menampilkan data

diabetes.head()

Hasil program :

| **Kehamilan** | **Glukosa** | **Tekanandarah** | **Tebalkulit** | **Insulin** | **BMI** | **FungsiPedigree** | **Umur** | **Luaran** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

**Langkah 3.** Memperhatikan dimensi data

print("Dimensi data diabetes : {}".format(diabetes.shape))

Hasil program

Dimensi data diabetes : (768, 9)

**Langkah 4.** Untuk menghitung tiap klas

#Untuk mengidentifikasi berapa diabetes (1 diabetes dan 0 jk tidak)

diabetes.groupby('Luaran').size()

Hasil program :

Luaran

0 500

1 268

dtype: int64

**Langkah 5.** Menampilkan data per faktor dengan berdasarkan klasifikasi Luaran

diabetes.groupby('Luaran').hist(figsize=(9, 9))

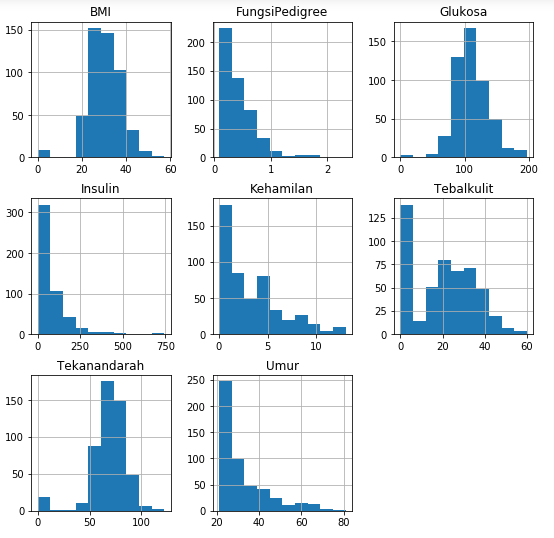
Hasil Program

Luaran

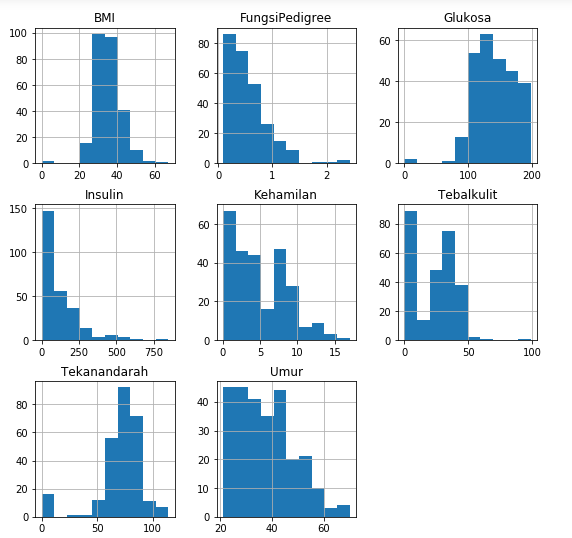
0 [[AxesSubplot(0.125,0.670278;0.215278x0.209722...

1 [[AxesSubplot(0.125,0.670278;0.215278x0.209722...

dtype: object



Gambar 4. Distribusi data pasien yang berlabel 0 (tidak diabetes)



Gambar 5. Distribusi data pasien yang berlabel 1 (diabetes)

Penjelasan :

Pehatikan bahwa Gambar 4 menunjukkan distribusi data untuk masing-masing faktor. Perhatikan pada Ggambar 4 baris 1 kolom 1 menunjukkan distribusi BMI

* 1. **Membersihkan data**

Beberapa proses membersihkan data dikarenakan :

1. Observasi ganda ataupun tidak relevan.
2. Salah dalam pelabelan , kategori yang sama terjadi berulang.
3. Data hilang atau ada data yang kosong (null data)
4. Terdapat outlier yang tidak diharapkan.

Pada kegiatan ini dapat dianggap bahwa tentang bagian a atau b sudah baik (observasi relevan dan tidak salah dalam pelabelan). Untuk itu kita mengobservasi ada tidaknya data yang hilang.

**4.3.1 Mencermati data yang hilang**

Langkah 6. #Untuk mengenali ada tidaknya missing data, ternyata hasil menunjukkan tidak adanya data yang kosong

diabetes.isnull().sum()

diabetes.isna().sum()

Hasil program

Kehamilan 0

Glukosa 0

Tekanandarah 0

Tebalkulit 0

Insulin 0

BMI 0

FungsiPedigree 0

Umur 0

Luaran 0

dtype: int64

Penjelasan : tidak ada data yang missing . Akan tetapi ketika data ditelusur, terdapat kesalahan karena seseorang tidak mungkin mempunyai tekanan darah 0. Dengan menelusur data, maka terdapat 35 orang yang terhitung bertekanan darah 0.

**4.3.2 Mencermati data outlier**

Dengan memperhatikan histogram pada Gambar 1 dan 2 terdapat beberapa data bertipe outlier (tidak mengumpul pada sebagian besar data). Data yang memuat outlier adalah data pada tekanandarah, level glukosa, ketebalan kulit, BMI dan insulin. Setelah teramati, kemudian dapat ditentukan bagaimana data diolah selanjutnya.

Langkah 7. Perintah untuk menelusur banyaknya data bertekanan darah 0

print("Total : ", diabetes[diabetes.Tekanandarah == 0].shape[0])

Hasil program :

('Total : ', 35)

Luaran

0 19

1 16

Name: Umur, dtype: int64

Langkah 8. Mencermati level glukosa plasma : sekalipun sudah puasa, level glukosa tidak mungkin 0 sehingga jika level glukosa bernilai 0 berarti terdapat kesalahan data. Dengan mengobservasi data, maka diperoleh 5 orang mempunyai nilai glukosa 0

print("Total : ", diabetes[diabetes.Glukosa == 0].shape[0])

Hasil Program

('Total : ', 5)

Langkah 9. Dari 5 orang di atas akan dicermati berapa orang yang teridentifikasi diabetes (berlabel 1) dan yang tidak diabetes (berlabel 0)

print(diabetes[diabetes.Glukosa == 0].groupby('Luaran')['Umur'].count())

Hasil program

Luaran

0 3

1 2

Name: Umur, dtype: int64

Langkah 10. Demikian pula kita akan mencermati berdasarkan Tebal kulit dimana orang normal ketebalan bawah kulit tidak boleh lebih kecil dari 10 mm. Dari program diperoleh bahwa yang bernilai 0 sebanyak 227.

Program

print("Total : ", diabetes[diabetes.Tebalkulit == 0].shape[0])

print(diabetes[diabetes.Tebalkulit == 0].groupby('Luaran')['Umur'].count())

Hasill program :

('Total : ', 227)

Luaran

0 139

1 88

Name: Umur, dtype: int64

Penjelasan : Jadi dari 227 orang teridentifikasi tebal kulit 0 maka untuk orang yang tidak teridentifikasi diabetes (berlabel 0) ada 139 dan yang berlabel 1 ada 88 orang).

Langkah 11. Demikian pula untuk BMI, tidak boleh teridentifikasi BMI bernilai 0 , untuk itu dihitung berapa orang BMI bernilai 0 dan berdasarkan klasnya maka dapat teridentifikasi dengan 9 orang yang tidak diabetes dengan BMI bernilai 0 dan 2 orang BMI dengan nilai 0 untuk 2 orang diabetes.

#BMI tidak boleh lebih kecil dari nol

print("Total : ", diabetes[diabetes.BMI == 0].shape[0])

print(diabetes[diabetes.BMI == 0].groupby('Luaran')['Umur'].count())

Hasil program

Outcome

0 9

1 2

Name: Age, dtype: int64

Langkah 12. Secara sama , kita dapat mengidentifikasi kadar insulin pada tiap klas. Pada situasi sangat jarang, seseorang dapat mempunyai insulin 0 tetapi berdasarkan data observasi maka terdapat 374 orang dinyatakan berkadar insulin 0.

**Program :**

print("Total : ", diabetes[diabetes.Insulin == 0].shape[0])

print(diabetes[diabetes.Insulin == 0].groupby('Luaran')['Umur'].count())

Hasil program :

('Total : ', 374)

Luaran

0 236

1 138

Name: Umur, dtype: int64

Langkah 13. Untuk mengedit data

Data –data diatas dibaca dan dikatakan invalid. Untuk mengatasi data invalid, kita akan melakukan pembersihan data dengan menghapus faktor Tekanan darah, BMI dan Glukosa yang bernilai 0 pada program berikut

diabetes\_mod = diabetes[(diabetes.Tekanandarah != 0) & (diabetes.BMI != 0) & (diabetes.Glukosa != 0)]

print(diabetes\_mod.shape)

Hasil program :

(724, 9)

Penjelasan : Jadi semula data observasi memuat 768 orang, maka data dibuang menjadi 724 dimana data yang dibuang memuat data yang invalid.

* 1. **Proses Teknis lebih lanjut**

Karena data sudah disediakan dan dengan mencermati data seperti di atas, kita tidak dapat membuat atau menghilangkan data pada langkah ini. Pada data yang sudah ada maka kita mempunyai 9 faktor diatas. Dengan observasi kasar tanpa program, ketebalan kulit sebenarnya bukan indicator dalam diabetes akan tetapi kita tidak dapat menghindari fakta bahwa data tersebut sebenarnya tidak berguna pada langkah ini. Oleh karena itu kitaakan menggunakan semua faktor yang sudah ada dan kita akan memisahkan data dimana data terdiri dari variabel X dan Y.

Program :

namafaktor = ['Kehamilan', 'Glukosa', 'Tekanandarah', 'Tebalkulit', 'Insulin', 'BMI', 'FungsiPedigree', 'Umur']

X = diabetes\_mod[namafaktor]

y = diabetes\_mod.Luaran

Umumnya faktor teknis ditunjukkan sebelum pemilihan model. Akan tetapi pada makalah ini kita melakukan pendekatan yang berbeda. Faktor teknis (ditulis dalam bahasa Inggris *adalah Feature Enginerring* ) .Pendekatan tersebut yaitu mencermati semua faktor dalam data dan mendiskusikan seberapa penting faktor-faktor tersebut dalam model.

* 1. **Pemilihan Model**

### Pemilihan model atau algoritma pemilihan merupakan algoritma yang paling menarik dan menjadi jantung utama dalam *machine learning*. Hal ini merupakan fase dimana kita memilih model dan menampilkan data secara terbaik. Langkah awal yang dilakukan adalah melakukan Uji Akurasi dari model-model klasifikasi berdasarkan parameter yang sudah ditetapkan untuk tiap model dikatakan baik. Algoritma utama untuk hal ini adalah **K-Nearest Neighbors, Support Vector Classifier, Logistic Regression, Gaussian Naive Bayes, Random Forest and Gradient Boost**  sebagai pengklasifikasi model terbaik.

**Langkah 14**. Mengimport algoritma pengklasifikasi model

Program

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

**Langkah 15**. Kita akan menginisialiasi model-model klasifikasi dengan parameter yang sudah baku (default) dan menambahkan parameter tersebut dalam daftar model

Program

models = []

models.append(('KNN', KNeighborsClassifier()))

models.append(('SVC', SVC()))

models.append(('LR', LogisticRegression()))

models.append(('DT', DecisionTreeClassifier()))

models.append(('GNB', GaussianNB()))

models.append(('RF', RandomForestClassifier()))

models.append(('GB', GradientBoostingClassifier()))

**4.5.1 Mengevaluasi Model**

Model mempunyai tujuan untuk memprediksi data yang diluar sampel data dan model dapat kompleks sehingga dapat terjadi *overfitting*.

Untuk menghindari hal-hal demikian, terdapat 2 langkah awal :

1. Memisahkan data menjadi data latih dan data uji yang disebut train/test method
2. Melakukan validasi silang K-kali yang disebut K-Fold Cross Validation

Selanjutnya algoritma akan memberikan nilai cross\_val\_score dari langkah ke-2 di atas untuk mengevaluasi akurasi dari model dalam metode train/test

**Langkah 16**. Mengimport fungsi-fungsi yang diperlukan

Program

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

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import KFold # ini ditambahkan krn tidk ada tetapi perlu

Selanjutnya kita akan menampilkan metode di atas untuk menentukan model standar yang terbaik.

* + 1. **Pemisahan data latih dan data uji**

Pada langkah ini kita akan membuat data menjadi 2 yaitu data latih (training set) dan data uji (testing set) dimana data uji untuk menguji model dan mengevaluasi akurasi.

**Langkah 17.** **Pemisahan dengan Scikit Learn**

Jadi kita akan memisahkan faktor-faktor dan respon dalam data latih dan data uji. Kita melakukan proses ini dimana setiap klas respon (misal yang berlabel diabetes) diwakili dengan porsi yang sama pada tiap sampel data.

Program

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, stratify = diabetes\_mod.Luaran, random\_state=0)

Selanjutnya kita akan memasangkan data terhadap setiap model dan menghitung akurasi setiap model dengan nilai akurasi.

**Program**

names = []  
scores = []

for name, model in models:  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
 scores.append(accuracy\_score(y\_test, y\_pred))  
 names.append(name)

tr\_split = pd.DataFrame({'Nama': names, 'Nilai': scores})  
print(tr\_split)

Hasil program :



Penjelasan : setiap model mempunyai nilai akurasi. Jika semua angka di belakang koma signifikan, maka model LR(*Logistic Regression*) dianggap paling akurat.

* + 1. **Validasi silang (K cross validation)**

Langkah 18.

. Proses ini akan dilakukan K-kali dengan menggunakan partisi yang berbeda sebagai data uji setiap waktu. Akurasi rata-rata uji dalam proses merupakan akurasi hasil uji model. Dengan ini estimasi untuk data diluar data sampel akan lebih akurat serta dikatakan lebih efisien dalam menggunakan data (setiap observasi digunakan untuk data latih dan data uji). Akan tetapi kendala yang dihadapi bahwa metode ini lebih lambat dibandingkan metode pemisahan di atas (Train /Test Split)

Proses ini digambarkan pada Gambar 6 .

Data

Data uji (test) (Te)

Data latih (*training*) (Ta)

(Te)

(Te)

Gambar 6. Partisi data dengan *K*=3 kali partisi

Program

names = []

scores = []

for name, model in models:

kfold = KFold(n\_splits=10, random\_state=10)

score = cross\_val\_score(model, X, y, cv=kfold, scoring='accuracy').mean()

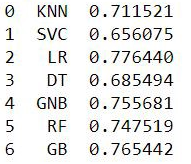
names.append(name)

scores.append(score)

kf\_cross\_val = pd.DataFrame({'Nama': names, 'Nilai': scores})

print(kf\_cross\_val)

Hasil program



Penjelasan : dengan *K*=10 dinyatakan pada variabel nsplit=10, maka diperoleh bahwa LR masih tetap sebagai kandidat model terbaik untuk memodelkan data yang sudah dibersihkan atau mempunyai 77.64 % akurasi.

Langkah 19. Bagian ini merupakan bagian yang menampilkan tabel akurasi dengan grafik dan kita menggunakan menu seabond untuk itu.

Program

axis = sns.barplot(x = 'Nama', y = 'Nilai', data = kf\_cross\_val)

axis.set(xlabel='Klasifikasi', ylabel='Akurasi')

for p in axis.patches:

tinggi = p.get\_height()

axis.text(p.get\_x() + p.get\_width()/2, tinggi + 0.005, '{:1.4f}'.format(tinggi), ha="center")

plt.show()

Hasil program

(belum)

1. **Kesimpulan**

Model dipilih adalah Logistic Regression (LR) Untuk itu data selanjutnya diolah menggunakan model ini. Pada beberapa materi sebelum ini pernah dilakukan pemodelan dengan LR. Oleh karena itu, sebelum melakukan LR kita harus menyimpan data yang sudah dibersihkan dalam format \*csv yang baru untuk dilakukan pemodelan.

1. **Catatan dan Saran**

Perhatikan bahwa dalam penulisan di atas kita hanya belajar dalam membaca data, membersihkan data dan mengevaluasi data terhadap beberapa model sehingga dapat dipilih secara otomatis atas model-model yang dipilih. Terdapat banyak model yang sudah diujicobakan terhadap data tersebut dimana kita belum menelusur masing-masing model secara lebih detail, yaitu **K-Nearest Neighbors, Support Vector Classifier, Logistic Regression, Gaussian Naive Bayes, Random Forest and Gradient Boost** akan tetapi telah dipelajar melalui video. Pemilihan data terhadap model Logistic Regression menunjukkan bahwa model yang lain dapat *underfitting* atau *overfitting*. Selain memahami masing-masing model lebih detail, maka hasil pemodelan terhadap data dengan LR perlu diperhatikan lebih lanjut.

1. **Proses Pemodelan data yang sudah bersih**

Sebagaimana disebutkan bahwa data telah dibersihkan dimana dalam program disebut sebagai diabetes\_mod

Kita dapat langsung memproses pengolahan data pada program yang baru dengan hasil identifikasi digunakan untuk langsung kembali memanggil data

Misalkan file program diberi nama Diabetesbersih.ipynb

Langkah 20

#https://towardsdatascience.com/machine-learning-workflow-on-diabetes-data-part-01-573864fcc6b8

%matplotlib inline

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

diabetes = pd.read\_csv('E:/JAN2016/KULIAH20182019/SEM2\_20182019/ANALISA\_DATA/PROJECTAnalisaData/diabetesIndo.csv')

diabetes.columns

Langkah 21. Memanggil data yang bersih saja setelah observasi di atas

diabetes\_mod = diabetes[(diabetes.Tekanandarah != 0) & (diabetes.BMI != 0) & (diabetes.Glukosa != 0)]

print(diabetes\_mod.shape)

Luaran

(724, 9)

Langkah 22. Menyumpannya dalam file :

#menyimpan data pada file Simpan. csv

diabetes\_mod.to\_csv('Cleandiabetes.csv')

Selanjutnya kita dapat melakukan logistic Regression

Web 1 . <https://machinelearningmastery.com/master-machine-learning-algorithms/>

Web 2. <https://yout.com/video/QKSvu3mj-14/>

Web 3. <https://www.coursera.org/learn/python-machine-learning>

Web 4.

<https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html>