BAB II. LANDASAN TEORI

2.1. Machine Learning

Berkenaan dengan penelitian kami mengenai pengelolaan infrastruktur Teknologi Informasi dalam hal memahami *kinerja server*, maka kami menawarkan penggunaan metode *Machine Learning*. *Machine Learning* dapat dipergunakan untuk menghasilkan prediksi serta memperbaiki sistem, dengan membuat keputusan yang lebih akurat, berdasarkan informasi (*execution, resources and requirement*) yang tersedia. *Machine Learning* menggunakan teknik berdasarkan catatan keadaan informasi masa lalu untuk kemudian membuat suatu model yang tepat atas keadaan yang sering terjadi, selanjutnya mengenali anomali sistem hingga menghasilkan keputusan serta evaluasi terhadap sistem tersebut *(Berral et al., 2012).*

Machine Learning merupakan teknik tata kelola infrastruktur Teknologi Informasi yang memungkinkan proses pengambilan keputusan dibuat dengan pemodelan terhadap sistem yang ada berdasarkan data-data sistem tersebut, sehingga model yang dihasilkan tersebut dapat diperbaharui, maupun membuat model baru lainnya yang disesuaikan dengan kebutuhan spesifik tertentu. Dengan kata lain, bahwa dengan penggunaan *Machine Learning* dalam tata kelola infrastruktur Teknologi Informasi, kita akan dapat menggali pengetahuan secara langsung atas perilaku sistem yang sedang berjalan *(Berral et al., 2012).*

Machine Learning dimulai dengan mengolah data-data yang ada, bertujuan untuk memperoleh informasi yang saling berhubungan dan untuk menentukan mana yang menjadi atribut, kemudian membuat suatu model yang

dapat dipergunakan untuk menjelaskan bagaimana kondisi sistem yang ada, hingga akhirnya dibuatlah suatu keputusan berdasarkan model tersebut.

Secara umum, cara kerja *Machine Learning* adalah dengan mengolah serangkaian data yang disebut sebagai *data set,* yang berasal dari suatu sistem dengan menentukan nilai-nilai dari sistem tersebut, menentukan mana yang atribut dan mana yang respon, kemudian membuat model berdasarkan nilai-nilai ini, sehingga ketika ada data yang baru, nilai yang diharapkan akan sesuai dengan ekspektasi atas model yang diperoleh.

Menurut *Josep LL. Berral* dan kawan-kawan *(Berral et al., 2012)*, teknik implementasi *Machine Learning* dibagi dalam beberapa pendekatan, diantaranya adalah :

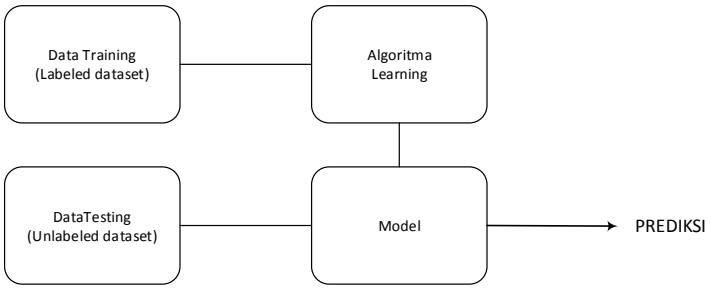
- *supervised learning*, seperti: *classification* maupun *regression*
- *unsupervised learning* (untuk mendapatkan pemahaman bagaimana relasi diantara data-data input), seperti: *clustering* (untuk mendapatkan pemahaman mengenai kemiripan atas data-data input) ataupun *reinforcement learning* (memilih keputusan yang paling baik berdasarkan kejadian masa lalu)

Dalam *supervised learning*, tujuannya adalah untuk memprediksi hasil berdasarkan input; sedangkan dalam *unsupervised learning*, hasil akhirnya bukanlah ukuran (*no outcome measure*), melainkan tujuannya adalah menjelaskan hubungan *(associations)* serta pola diantara data-data input. Meski untuk kedua pendekatan ini, sangat masuk akal menggunakan data input untuk membuat prediksi sebagai output *(Hastie, Tibshirani, & Friedman, 2009).*

2.1.1. Supervised Learning

Supervised Learning merupakan salah satu teknik dalam *Machine Learning* yang langsung mempelajari dari data operasional suatu sistem, yang memungkinkan dilakukannya prediksi dari sistem tersebut. Data untuk keperluan prediksi itu sendiri, terdiri dari berbagai elemen informasi yang dapat berasal dari berbagai sumber, yang seringkali justru tidak menggambarkan secara jelas dan mudah dipahami, juga terkadang tidak lengkap. Setelah dilakukan pemilahan atas data-data yang ada, sehingga menghasilkan informasi yang dapat diolah lebih lanjut, maka langkah selanjutnya adalah membuat model apakah *classifier model* atau *regression model.* Dari pemilihan model ini, maka informasi yang didapat akan berupa informasi yang dapat dipergunakan untuk membantu proses pengambilan keputusan. Dari pemilihan model yang membantu proses pengambilan keputusan inilah, selanjutnya dapat membantu dalam melakukan prediksi ataupun estimasi terhadap informasi tersebut, serta dapat memperlihatkan bagaimana relasi yang terjadi antara data yang diamati dengan keadaan sistem yang berjalan. Sehingga *Machine Learning* akan sangat membantu dalam memahami sistem yang sedang berjalan, dengan cara yang lebih baik *(Berral et al., 2012).* Untuk melakukan prediksi, diperlukan pemilihan algoritma prediksi yang tepat, agar ketika diolah dengan data training dari beragam variasi data atas beban yang ada, menghasilkan prediksi yang baik. Begitu pula, untuk data yang akan dijadikan data training maupun data test, diperlukan data yang baik untuk kedua jenis data tersebut. Artinya adalah, bila setelah dilakukan training data dan diperoleh prediksi yang mendekati nilai yang diperoleh saat test, maka dapat dikatakan bahwa model yang dipilih telah sesuai, karena akan

memenuhi pula untuk data-data baru di masa mendatang. Gambar 2.1 berikut, memperlihatkan skema dasar proses *Supervised Machine Learning (Berral et al., 2012)*:



Gambar 2.1. Skema Supervised Machine Learning

Tabel 2. 1. Penelitian yang menggunakan Machine Learning

No	Peneliti	Penelitian	Model Machine Learning
1	Roy, C., Moitra, S., Das, M., Srinivasan, S., & Malhotra, R. (2015). IT Infrastructure Downtime Preemption using Hybrid Machine Learning and NLP, *6*, 39–44. https://doi.org/10.15439/2015F400	Tata kelola Infrastruktur TI	Support Vector Machine, Random Forest
2	Gao, J., & Jamidar, R. (2014). Machine Learning Applications for Data Center Optimization. *Google White Paper*, 1–13.	Optimisasi DC	Neural Network
3	Huang, C.-J., Wang, Y.-W., Guan, C.-T., Chen, H.-M., & Jian, J.-J. (2013). Applications of Machine Learning to Resource Management in Cloud Computing. *International Journal of Modeling and Optimization*, *3*(2), 148–152. https://doi.org/10.7763/IJMO.2013.V3.256	Alokasi resource	Support Vector Regression (SVR), Genetic Algorthm (GA)
4	Torres Viñals, J. (2010). Resource management on Cloud systems with Machine Learning.	Tata kelola resource	Linear Regression, Decision Tree
5	Alonso, J., Torres, J., & Gavaldà, R. (2009). Predicting web server crashes: A case study in comparing prediction algorithms. *Proceedings of the 5th International Conference on Autonomic and Autonomous Systems, ICAS 2009*, 264–269. https://doi.org/10.1109/ICAS.2009.56	Monitoring dan Prediksi Memory dan CPU terhadap Web Application	Linear Regression, Decision Tree
6	Zhang, Q., Cherkasova, L., Mi, N., & Smirni, E. (2008). A regression-based analytic model for capacity planning of multi-tier applications. Cluster Computing, 11(3), 197–211.	Pengaruh dan Prediksi Memory	Regression
7	Kapadia, N. H., Fortes, J. A. B., & Brodley, C. E. (1999). Predictive Application-Performance Modeling in a Computational Grid Environment, 47–54.	Prediksi Kinerja	Nearest-Neighbour, Polynomial Linear Regression

Penelitian yang dilakukan oleh *Josep LL. Berral* dan kawan-kawan, turut mendukung penelitian kami, yang berbasis pada model grid. Karena model grid sudah terlebih dahulu ada sebelum adanya model cloud. Penelitian berbasis model grid maupun cloud lainnya , diantaranya diperlihatkan dalam tabel 2.1 berikut:

Persamaan dalam *supervised learning* dapat dituliskan secara sederhana sebagai berikut *(Hastie et al., 2009)*:

$$Y = f(X) + E \qquad ; E = error \qquad (2.1)$$

Fungsi matematis ini, untuk mempelajari fungsi $f(X)$ berdasarkan data input. Antara input dan output dilakukan pengamatan (observasi) atas *data training*, dengan persamaan T = (x_i, y_i), $i = 1, \ldots, N$. *Algoritma learning* salah satu input nilai observasi (*artificial*), menghasilkan persamaan output $f\hat{}(x_i)$. Selisih antara ouput original dan output observasi, menghasilkan hubungan antara $f\hat{}$ dan y, yaitu: $y_i - f\hat{}(x_i)$. Proses ini dikenal dengan *learning by example*. Dalam proses *learning* ini, diharapkan bahwa *output artificial* dan *output real*, mempunyai nilai yang berdekatan. Karena menandakan bahwa persamaan yang diperoleh berguna untuk data input prediksi.

Dalam membuat prediksi untuk satu atau banyak output atau variabel respon, $Y = (Y_1, \ldots, Y_m)$ yang ditentukan dari banyaknya data input atau variabel predictor $X^T = (X_1, \ldots, X_p)$. Untuk data input yang diperoleh dari data training yang ke - i, menjadi $x_i^T = (x_{i1}, \ldots, x_{ip})$ dan untuk outputnya (*response*) menjadi y_i . Adapun sampel data training untuk prediksi berupa $(x_1,y_1), \ldots, (x_N,y_N)$.

Beberapa persamaan matematis diatas, adalah persamaan matematis yang banyak ditemui dalam kaitannya dengan pemecahan masalah menggunakan metode *supervised learning*. Metode *supervised learning* akan menghasilkan

jawaban berupa $y_i^{\wedge}$ untuk sampel data training x_i , yang kemudian akan dilakukan uji coba atas beberapa jawaban tersebut, hingga diperoleh nilai yang tepat, setelah serangkaian uji coba data training dan data testing. Nilai yang tepat ini, dalam membuat prediksi, biasanya diperoleh dengan menggunakan *loss function L (y , $y^{\wedge}$)*. Bila *(X , Y)* merupakan variabel acak, dan kerapatan probabilitas gabungan *(joint probability density)* dilambangkan dengan P_r *(X , Y)*, maka supervised learning dapat dikatakan sebagai persoalan kerapatan estimasi *(density estimation problem)* dengan syarat kerapatan *(conditional density)* $P_r (Y \mid X)$.

2.1.2. Unsupervised Learning

Menurut *Trevor Hastie, Robert Tibshirani dan Jerome Friedman (Hastie et al., 2009)*, pemecahan masalah dengan *unsupervised learning,* antara lain adalah dengan menggunakan *Cluster Analysis*. *Cluster Analysis* disebut juga *data segmentation*. Tujuannya adalah membuat pengelompokan *(grouping)* atau segmentasi *(segmenting)* dari sekumpulan obyek kedalam kelompok yang lebih kecil *(subset)* atau *"cluster"*. Di dalam setiap cluster obyek-obyek akan lebih saling berkaitan erat, dibandingkan bila obyek-obyek tersebut dikaitkan antar cluster. Suatu obyek dapat berupa serangkaian data pengukuran, ataupun relasinya dengan obyek lainnya. *Cluster analysis*, juga dapat dipergunakan untuk membentuk gambaran statistik yang menggambarkan relasi suatu obyek dalam kelompok yang lebih kecil (subgroup). Untuk mendapatkan tujuan tersebut, diperlukan pengujian tingkat perbedaan antara obyek yang dipilih terhadap clusternya. Tujuan utama *Cluster analysis* adalah mendapatkan tingkat kemiripan (atau perbedaan) antara obyek individu dalam cluster. Teknik fundamental dalam

membuat *cluster* adalah memilih jarak atau perbedaan yang terukur antara dua obyek. Teknik tersebut antara lain adalah: *Proximity Matrices*, *Dissimilarities Based on Attributes , Object Dissimilarity*. Algoritma Cluster (*Clustering Algorithms*), dibagi menjadi: *combinatorial algorithms, mixture modeling,* dan *mode seeking*. *Combinatorial algorithms* bekerja secara langsung pada data observasi tanpa berkaitan langsung pada model probabilitasnya. M*ixture modeling* mengandaikan pada data sampel yang berasal dari suatu populasi yang merupakan fungsi kerapatan probabilitas (*probability density function*). Fungsi kerapatan ini merupakan komponen kerapatan yang menjelaskan masing-masing cluster nya. Untuk itu, model ini cocok digunakan untuk data dengan pendekatan Bayesian. *Mode seekers ("bump hunters")* merupakan perspektif nonparametrik, yang langsung melakukan estimasi dari fungsi kerapatan probabilitas. Observasi 'yang paling mendekati' mode nya, akan menentukan cluster individualnya.

Algoritma K-means, merupakan salah satu metode *iterative clustering* yang paling populer digunakan. Terutama bila semua variabel merupakan *tipe kuantitatif (Hastie et al., 2009).*

2.2. Penentuan Atribut dan Target

Menurut *Bowles, M. (Bowles, M., 2015),* algoritma *Machine Learning* dimulai dari mengumpulkan serangkaian data, mengamati data yang ada, menentukan komponen mana yang akan menentukan prediksi. Komponen tersebut dapat dibedakan menjadi:

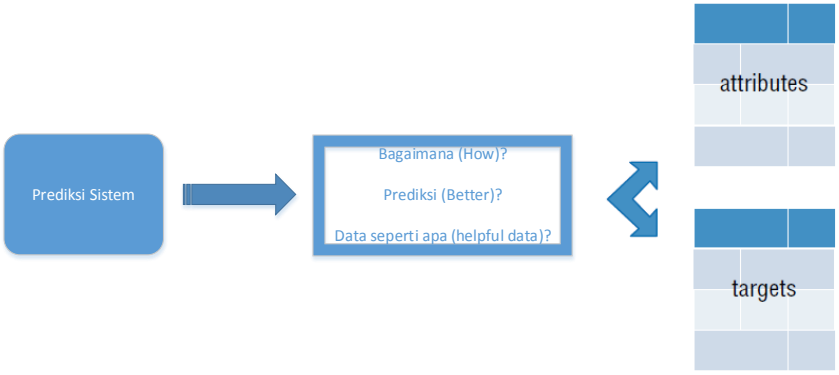
- *Atribut*, yaitu: variabel yang dipergunakan sebagai input dalam membuat prediksi. Dikenal juga dengan beberapa istilah lain, seperti: *predictor, feature,*

independent variables, input.

- *Target*, yaitu: hal yang diinginkan sebagai suatu prediksi.

  Dikenal juga dengan beberapa istilah lain, seperti: *outcomes, label, dependent variables, response*.

Dengan kata lain, *atribut* diperlukan untuk memprediksi *target*. Selengkapnya ditunjukkan dalam gambar 2.2 dibawah ini, *(Bowles, M., 2015):*



Gambar 2. 2. Komponen prediksi dalam Machine Learning

Selanjutnya, data akan diatur dalam bentuk baris dan kolom. Setiap baris mewakili satu kejadian tersendiri (disebut juga dengan istilah: *instance, example* ataupun *observation*). Sedangkan kolom mewakili problem yang menjadi *input (label)* dalam *Machine Learning*.

Atribut merupakan suatu hal yang dipilih untuk dipergunakan dalam penentuan prediksi. Sedangkan *target (label)* merupakan hasil observasi yang dipergunakan oleh algoritma *Machine Learning* dalam menyusun suatu model prediksi (*predictive model, (Bowles, M., 2015)*). Proses dalam membangun model prediksi ini, disebut sebagai *training*. Secara mendasar, algoritma proses training ini, memperlihatkan relasi antara atribut dan label dalam pembuatan model

prediksi, bagaimana kesalahan terjadi, melakukan perbaikan/koreksi atas kesalahan tersebut. Proses ini terus berulang hingga mendapatkan model yang tepat. Inilah ide dasar pembuatan suatu model prediksi.

Pemilihan fitur (*feature selection*) merupakan proses, diantara data-data yang diamati, dalam menemukan variabel bagi atribut tersebut dan menentukan data-data mana yang benar-benar berguna, dengan memisahkannya dengan data yang tak perlu atau tak mendukung penelitian (*noise* atau *irrelevant, (Berral et al., 2012)*). Seperti persoalan optimalisasi atas konfigurasi sistem, harus diketahui mengenai bagaimana aplikasi berjalan maupun yang harus dipenuhi aplikasi tersebut untuk berjalan baik, bagaimana utilisasi sumber daya diatur dalam mendapatkan konfigurasi terbaik dan kinerja terbaiknya. Maka metode pemilihan fitur sangat menentukan dalam mendapatkan atribut yang tepat yang berguna dalam memecahkan masalah.

2.3. Menguji (beberapa) model prediksi

Begitu pula dalam memahami sistem yang sedang berjalan dan menginginkan bagaimana prediksi atas sistem tersebut, maka perlu memahami bagaimana membuat sistem lebih optimal serta konfigurasi seperti apa yang diperlukan dalam menjalankan suatu aplikasi, yang sebanding dengan utilitas dari sumber daya yang ada, dalam hal mendapatkan konfigurasi terbaik dan kinerja terbaiknya. Hal inilah yang menjadi perhatian pada saat pemilihan fitur, menemukan atribut yang berguna dalam kaitannya dengan prediksi kinerja infrastruktur.

Pengujian kinerja *(performace assessment)* atas suatu model prediksi, dapat dilakukan dengan menggunakan *Loss Function (Hastie et al., 2009),*

$$L(Y, \hat{f}(X)) = \sum (Y - \hat{f}(X))^2 \text{ ; squared error} \quad (2.2)$$

$$L(Y, \hat{f}(X)) = \sum |Y - \hat{f}(X)| \text{ ; absolute error} \quad (2.3)$$

Catatan: $L(Y, \hat{f}(X))$ *: loss function = mengukur error antara Y dan* $\hat{f}(X)$

Y *: variabel target*

X *: variabel input*

$\hat{f}(X)$ *: model prediksi, diperoleh dari data training*

2.4. Pemilihan Model Prediksi

Setelah menentukan data mana yang menjadi komponen atribut dan mana yang menjadi komponen target, maka selanjutnya model prediksi ditentukan berdasarkan jenis atribut dan target tersebut.

Untuk komponen *atribut*, dibedakan menjadi:

- *Atribut dengan variabel numerik*
- *Atribut dengan variabel kategorial atau faktor*

Untuk komponen *target*, dibedakan menjadi:

- *Target numerik, maka persoalannya menjadi regression.*
- *Target kategorial, maka persoalannya menjadi classification.*

Meski persoalan numerik (*regression*) dapat di konversi menjadi persoalan kategorial (*classification*), begitu juga sebaliknya. Tentu hal ini akan disesuaikan dengan kebutuhan dan perancangan dalam pemecahan persoalan. Namun dikatakan oleh *Bowles M*, persoalan *classification* biasanya lebih sederhana dibandingkan dengan persoalan *regression (Bowles, M., 2015. p27).*

Pengujian kinerja *(performace assessment)* atas suatu model prediksi dilakukan, untuk selanjutnya dipergunakan dalam memilih model prediksi. Pengertian menurut (Hastie et al., 2009), dalam memahami antara pemilian model dan pengujian model adalah sebagai berikut:

- pemilihan model (*model selection*): model yang berbeda-beda, kemudian dipilih model dengan kinerja yang terbaik.
- Pengujian model (*model assessment*): prediksi error dengan menggunakan data baru.

2.5. Data Training dan Data Test

Setelah fitur (atribut) dan label (target) dipilih, proses selanjutnya adalah melakukan *training* pada model. Untuk itu, sampel data harus sampel yang baik, dalam arti data yang memiliki *noise* yang minimal dan nilai ganda juga seminimal mungkin. Menurut *Blum AL (Blum & Langley, 1997),* yang memberi contoh bagaimana untuk *n fitur* atau *n atribut* dipergunakan dalam menerangkan sampel dan masing-masing fitur *i* dalam domain F_i. Dijelaskan bahwa suatu fitur dapat berupa *Boolean (is_red?)*, atau diskrit bernilai banyak (*discrete with multiple value (what_color?)*, maupun kontinyu *(what_wavelength?)*. Sedangkan sampel merupakan salah satu poin dalam *instance space* $F_1 \times F_2 \times ... \times F_n$. Algoritma

learning diberikan dalam serangkaian *S training data*, dengan setiap data point merupakan sampel yang dipasangkan dengan *label* atau *classification* yang sesuai (*associated label or classification),* yang dapat berupa *Boolean*, *multiple value*, atau *continuous*.

Sedangkan menurut *Trevor Hastie, Robert Tibshirani, Jerome Friedman (Hastie et al., 2009)*, pendekatan terbaik dalam melakukan pemilihan model (*model selection*) dan pengujian model (*model assessment*), adalah dengan cara membagi secara acak dataset yang ada menjadi tiga bagian, yaitu:

- 50 % : data training, untuk mencocokan data dengan model *(to fit the model).*
- 25 % : data validation, untuk meng-estimasi prediksi error atas suatu model yang dipilih *(to estimate prediction error for model selection).*
- 25 % : data test, untuk menguji error secara umum atas model final yang dipilih *(to assessment generalized error of the final model).*

Adapun metode dalam melakukan pemilihan model (*model selection*) dan pengujian model (*model assessment*), adalah dengan penggunaan ulang sampel *(efficient sample re-use)*: *cross-validation* ataupun *bootstrap*.

2.6. Metode cross validation

Metode resampling adalah tool yang sangat diperlukan dalam penggunaan statistik untuk suatu penelitian. Metode resampling tersebut melakukan berulang kali pengambilan sampel dari suatu dataset training dan memperbaiki model di setiap sampel untuk mendapatkan informasi tambahan hingga diperoleh model yang baik. Sebagai contoh, untuk memperkirakan variabilitas atas model linear

regresi, maka dapat dilakukan berulang kali pengambilan sampel yang berbeda dari data training, kemudian melakukan fitting atas model regresi linier untuk setiap sampel baru, dan kemudian memeriksa mana yang menghasilkan model yang lebih baik. Pendekatan semacam itu memungkinkan kita dalam mendapatkan informasi yang tidak akan tersedia ketika model menggunakan sampel data training yang asli dan dilakukan sekali saja (James, Witten, Hastie, & Tibshirani, 2013).

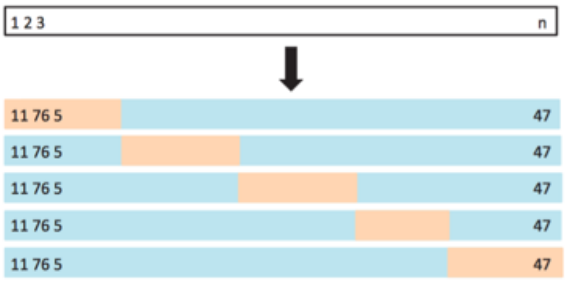
Metode resampling yang paling umum digunakan adalah cross-validation dan bootstrap. Kedua metode tersebut merupakan tool penting dalam penelitian menggunakan statistik. Metode Cross validation dapat digunakan untuk memperkirakan score error yang terkait dengan metode statistik tertentu untuk mengevaluasi kinerjanya, atau untuk memilih tingkat fleksibilitas (*level of flexibility*) yang sesuai. Proses mengevaluasi kinerja model dikenal sebagai penilaian model (*model assessment*), sedangkan proses pemilihan tingkat fleksibilitas yang tepat untuk model dikenal sebagai pemilihan model (*model selection*).

Bootstrap digunakan dalam beberapa konteks, yang paling umum adalah dalam memberikan ukuran akurasi estimasi parameter atau pilihan metode statistik.

Cross validation dijelaskan sebagai berikut: misalkan kita ingin memperkirakan kesalahan uji (test error) yang terkait dengan fitting suatu model menggunakan metode statistik pada suatu observasi. Validasi ini secara acak membagi kumpulan observasi yang ada menjadi dua bagian, satu bagian menjadi *training set* dan satu bagian lainnya menjadi *validation set* atau *hold-out set*.

Model di lakukan fitting pada data training, dan selanjutnya model yang telah di fitting tadi digunakan untuk memprediksi respon menggunakan data validasi. Validasi yang dihasilkan-biasanya menggunakan MSE dalam persoalan kuantitatif - menghasilkan perkiraan tingkat kesalahan uji (test error).

Metode cross validation pun dapat dilakukan dengan cara LOOCV (Leave One Out CV) atau k-fold CV. Penelitian ini menggunakan k-fold CV. Gambar 2.3 berikut memperlihatkan algoritma k-fold cross validation



Gambar 2. 3. k-fold Cross Validation: Data observasi yang dibagi menjadi data training dan data validasi

Untuk melakukan k-fold cross validation, dilakukan dengan membagi (split) data observasi secara acak menjadi k-bagian (k-fold), lalu:

i. pilih satu bagian (one-fold) sebagai validation set
ii. fitting pada model untuk bagian data lainnya (k-1 fold)
iii. ulangi langkah diatas hingga nilai k tercapai
iv. hitung score CV menggunakan: $CV_k = \frac{1}{k} \sum_{i=1}^{k} Error_i$

2.7. Pengaruh Bias – Variance

Bias dan variance adalah hal yang perlu diperhatikan dalam memperhitungkan prediksi menggunakan machine learning. Seperti telah dituliskan diatas, bila persamaan suatu prediksi dalam machine learning adalah (*James, Witten, Hastie, & Tibshirani, 2013*):

$$Y = f(X) + E \qquad (2.4)$$

Bila E(E) = 0 dan Var (E) = τ_E^2, maka diperoleh persamaan prediksi error *(expected prediction error)* dari pendekatan regression dengan $\hat{f}(X)$ pada suatu titik input X = x_0 , dengan menggunakan *squared-error loss,* (*James et al., 2013*):

$$\begin{aligned} \text{Err}(x_0) &= E[(Y - \hat{f}(x_0))^2 \mid X = x_0] \\ &= \tau_E^2 + [E\hat{f}(x_0) - f(x_0)]^2 + E\,[\hat{f}(x_0) - E\hat{f}(x_0)]^2 \\ &= \tau_E^2 + \text{Bias}^2(\hat{f}(x_0)) + \text{Var}(\hat{f}(x_0)) \end{aligned} \qquad (2.5)$$

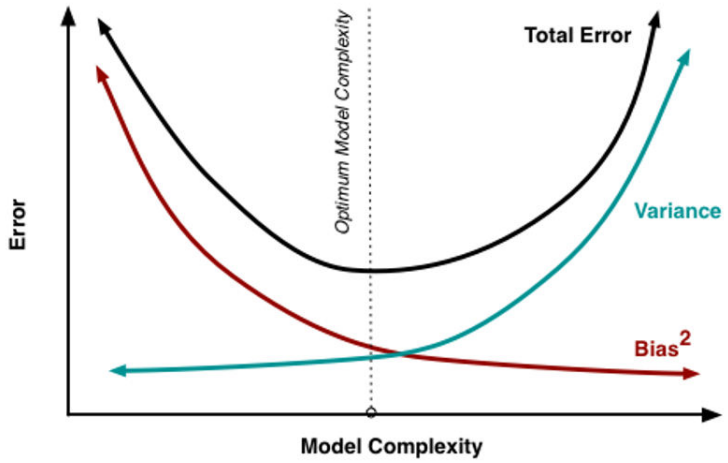
$$= \textit{Irreducible Error} + \textit{Bias}^2 + \textit{Variance}$$

Catatan:

- *Irreducible Error*, merupakan *target variance* di sekitar nilai rata-rata $f(x_0)$, dan merupakan nilai error yang tidak dapat dihindari, walau sebaik apapun kita melakukan estimasi atas $f(x_0)$, kecuali $\tau_E^2 = 0$.

- *Bias*^{2} , bias kuadrat, merupakan nilai estimasi rata-rata yang berbeda dari nilai rata-rata sesungguhnya.
- *Variance* , merupakan nilai kuadrat simpangan (*expected squared deviation*) $\hat{f}(x_0)$ disekitar nilai rata-rata nya.

Secara umum, semakin kompleks pembuatan model $\hat{f}$, maka semakin rendah bias (kuadrat), namun makin tinggi variance nya.



Gambar 2. 4. Kurva Bias – Variance Trade off (James et al., 2013)

Gambar 2.4, memperlihatkan total error (kurva hitam) terhadap kurva fleksibilitas (kompleksitas) model. Makin tinggi kompleksitas model, maka makin tinggi pula variance nya (kurva hijau), namun bias semakin kecil (kurva merah).

2.8. Efektivitas Banyaknya Parameter

Seberapa banyak parameter dibutuhkan dalam pembuatan suatu model prediksi, ditentukan oleh kesesuaian *(fitting)* model tersebut. Maka metode *linear fitting* dapat dituliskan sebagai berikut *(Hastie et al., 2009)*:

$$\hat{y} = Sy \qquad (2.6)$$

bila, y : merupakan hasil output $y_1, y_2, \ldots, y_N$ berupa vektor y, demikian pula untuk prediksi $\hat{y}$. S merupakan matrik $N \times N$, tergantung pada vektor data input x_i namun tidak tergantung pada y_i . Selanjutnya, banyaknya parameter yang efektif dinyatakan dalam (Hastie et al., 2009):

$$\text{df}(\mathbf{S}) = \text{trace}(\mathbf{S}) \qquad (2.7)$$

dalam arti: yang merupakan nilai dari jumlah elemen diagonal dari **S** (yang juga dikenal sebagai *effective degrees-of-freedom*).

2.9. Penggunaan Virtualisasi

Teknologi virtualisasi merupakan salah satu kunci penting dalam pengelolaan infrastruktur Teknologi Informasi yang memungkinkannya pengurangan biaya produksi (cost reduction) maupun kemudahan dalam pengelolaan sumber daya seperti penyedia jasa umumnya (Berral et al., 2012). Berkat penggunaan teknologi virtualisasi, beberapa proses, juga banyak guest operating system (OS) maupun VM dapat berjalan dalam satu physical machine maupun banyak physical machine dalam berbagai platform berbeda. Yang memungkinkan konsolidasi berbagai aplikasi dengan pemanfaatan sumber daya infrastruktur fisik. Juga dengan teknologi virtualisasi ini, komponen tertentu dalam aplikasi (seperti 'task') dapat dijalankan di mana saja dan dapat dimigrasi tanpa mengalami banyak kendala, bahkan VM juga dapat ditingkatkan / dioptimalkan kinerjanya baik melalui host OS maupun physical machine nya.

Tujuan utama virtualisasi adalah menyiapkan batasan (confined) bagi lingkungan (environment) infrastruktur agar suatu aplikasi dapat berjalan dengan baik, menetapkan batas (limit) terhadap akses dan pemakaian sumberdaya

hardware, maupun perluasan (expand) pengunaan sumber daya hardware tersebut secara transparant bagi aplikasi tersebut, menyesuaikan waktu eksekusi (runtime environment) yang diperlukan bagi suatu aplikasi, penggunaan secara khusus (dedicated) atau melalui mekanisme optimalisasi OS untuk setiap aplikasinya, pengelolaan atas seluruh aplikasi yang ada dan memproses berjalannya aplikasi-aplikasi tersebut yang terdapat didalam VM.

Penelitian yang dilakukan oleh Primet dan kawan-kawan, mengenai solusi virtualisasi (*virtualization solution for grids*), disimpulkan sebagai berikut (Berral et al., 2012):

- *Pendekatan level-OS (OS-Level Approaches).* Melalui pendekatan ini, virtualisasi pada physical server memungkinkan banyak server virtual berjalan pada satu physical server. Tidak dipergunakannya OS di guest dan aplikasi berjalan dengan OS yang ada seakan-akan aplikasi tersebut berjalan sendiri pada OS tersebut. Contoh yang menggunakan pendekatan ini adalah Linux Vserver (Berral et al., 2012), suatu patch pada kernel berbasis partitioning, dengan menggunakan 'security context' didalam OS UNIX, FreeBSD Jail dan juga Solaris Containers, OpenVZ, dan lainnya.
- *Emulators*. VM mensimulasi hardware secara utuh dalam guest OS. Vmware merupakan software virtualisasi untuk mesin berbasis arsitektur x86. Virtualisasi sendiri bekerja di level processor, kemudian dengan perintah khusus (privileged instructions) yang telah ditanam dan divirtualisasi menggunakan proses Vmware dan beberapa perintah lainnya dapat langsung dieksekusi melalui processor di host. Seluruh sumber daya hardware dari mesin yang terbentuk juga dapat di virtualisasi. Solusi

virtualisasi lainnya adalah Microsoft VirtualPC, VirtualBox, QEMU, dan lainnya.

- *OS di sisi user (OS in User Space)*. Pendekatan ini dengan menyiapkan virtualisasi melalui guest OS secara langsung di sisi user. Beberapa pendekatannya seperti User Mode Linux, yang memungkinkannya berjalannya OS Linux sebagai suatu aplikasi pada mesin sebagai host yang berjalan dengan menggunakan Linux, hal yang sama juga untuk coLinux, Adeos, L4Ka-based projects, dan lainnya.
- *Paravirtualization*. Teknik paravirtualisasi tidak memerlukan proses mensimulasi hardware, namun sebagai penggantinya dengan menggunakan API khusus yang memodifikasi ke guest OS. Adapun sumber daya hardware nya pun merupakan sumber daya yang abstrak (abstract resources) yang tidak perlu sama dengan sumber daya hardware yang aktual dari mesin host nya. Xen merupakan VM monitor untuk arsitektur x86, yang memungkinkannya berjalan secara bersamaan beberapa OS namun tetap menyediakan sumber daya yang terisolasi dan terbatas selama menjalankannya diantara OS tersebut. Penggunaan pendekatan paravirtualisasi lainnya adalah seperti Denali dan Trango.
- *Hardware-Assisted Virtualization.* Teknik virtualisasi ini memungkinkan untuk menjalankan guest OS tanpa modifikasi (unmodified guest OS), sehingga membuat VM memiliki hardware nya sendiri. Hal ini dapat terjadi karena bertambahnya kemampuan instruksi yang dapat diolah oleh processor, yang dibuat oleh Intel VT (IVT), AMD (AMD Pacifica x86

virtualization), IBM (IBM Advanced POWER virtualization), dan SUN (Sun UltraSPARC T1 hypervisor).

Teknologi virtualisasi telah menjadi penelitian dalam menghasilkan keuntungan yang maksimal, namun juga telah menambah lapisan abstrak lainnya bagi sistem pengelolaannya, membuat makin rumitnya pengelolaan energi konvensional dalam rangka menghasilkan efisiensi maupun penyediaan lingkungan virtual yang tepat. Teknologi virtualisasi juga telah membuka penelitian yang lebih luas lagi dengan kemampuannya melakukan optimalisasi tata kelola infrastruktur baik secara cloud maupun grid (optimize cloud and grid management). Bagaimana kemampuannya dalam mengisolasi beberapa tugas (job) didalam VM, dan kemampuan migrasi VM dalam mesin fisiknya, juga memungkinkannya pengaturan tugas kecil (task) agar lebih optimal serta penjadwalan secara dinamik tanpa harus mengakibatkan atau mengganggu sistem. Saat ini, teknik machine learning telah diterapkan untuk membantu mengelola platform virtual dalam memutuskan penjadwalan VM, mengumpulkan informasi mengenai sistem secara menyeluruh sehingga dapat dibuat prediksi dengan pemanfaat informasi tersebut. Juga dengan adanya machine learning tersebut, dapat digunakan untuk mendapatkan pola mengenai perilaku VM dan sistem hostnya untuk memprediksi baik untuk jangka pendek maupun untuk jangka panjang, sehingga membantu dalam membuat keputusan maupun kebijakan secara lebih akurat dan untuk kurun waktu yang panjang.

2.10. Tinjauan Pustaka

Berkaitan dengan pengelolaan sumber daya (*resource allocation*) infrastruktur teknologi informasi, dalam rangka memaksimalkan utilisasi/kinerja komputasi (*maximum computing performance*) dan membuat infrastruktur yang tetap ramah lingkungan (*green computing*), serta pertimbangan biaya yang berkaitan dengan sumber daya yang disediakan *(Huang et al., 2013*), maka pembuatan model dilakukan agar pengelolaan sumber daya infrastruktur dapat memberikan kinerja yang optimal dan tidak berlebihan dalam memanfaatkan sumber daya infrastruktur tersebut. Penelitian dalam thesis ini mendasarkan pendekatan pada penelitian yang berbasis data (*data-driven*) dan dipergunakan untuk membantu menyelesaikan masalah serta membantu dalam pengambilan keputusan terutama yang berkaitan dengan komputasi, dengan cara membuat perancangan suatu sistem yang mengoptimalkan pengelolaan/alokasi sumber daya (*resource allocation*). Optimalisasi pengelolaan ini, dilakukan dengan membuat suatu prediksi atas data aktual/eksisting.

2.10.1. Model Linear Regression

Dengan kemunculan disiplin ilmu machine learning maupun disiplin ilmu lainnya, statistical learning telah muncul sebagai subfield baru dalam statistik yang difokuskan pada pemodelan dan prediksi supervised maupun unsupervised model (James, Witten, Hastie, & Tibshirani, 2013). Masih menurut Trevor Hastie dan Robert Tibshirani, dalam bukunya ‘An Introduction to Statistical Learning’, bahwa linear regression merupakan pendekatan yang sederhana untuk model supervised learning. Secara khusus, linear regression merupakan model yang

berguna untuk memprediksi sebuah respon kuantitatif. Model linear regression masih merupakan model yang sangat berguna dan banyak dipergunakan secara luas dalam metode statistical learning. Model linear regression merupakan model yang memiliki hubungan secara linier dari beberapa parameter dan memang lebih mudah dibuat daripada model yang memiliki hubungan yang non-linier. Secara spesifik, untuk data yang diolah berupa data numerik, linear regression merupakan pilihan yang alami (*natural method*) untuk dipertimbangkan (Torres Viñals, 2010).

Persamaan linier dengan banyak atribut (feature), dapat dituliskan sebagai berikut:

$$y = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k \quad (2.8)$$

y : merupakan output

$a_1, a_2, \ldots, a_k$: merupakan atribut/feature

$w_0, w_1, \ldots, w_k$: merupakan koefisien

Nilai koefisien diperoleh dari data training. Persamaan diatas, bila dituliskan dalam bentuk vektor untuk n-observasi, menjadi:

$$Y = AW + \varepsilon \quad (2.9)$$

dengan

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ .. \\ y_n \end{bmatrix}, A = \begin{bmatrix} A'_1 \\ A'_2 \\ .. \\ A'_n \end{bmatrix} = \begin{bmatrix} 1 & a_1^1 & \ldots & a_k^1 \\ 1 & a_1^2 & \cdots & a_k^2 \\ 1 & \vdots & \ldots & \vdots \\ 1 & a_1^n & \cdots & a_k^n \end{bmatrix}, W = \begin{bmatrix} w_0 \\ w_1 \\ .. \\ w_k \end{bmatrix}, \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ .. \\ \varepsilon_n \end{bmatrix} \quad (2.10)$$

Y : variabel dependent

A : variabel independent

W : koefisien regression

E : nilai error

Adapun nilai prediksi dirumuskan sebagai:

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(i)} \quad (2.11)$$

Sedangkan selisih antara nilai aktual (data test) dan nilai prediksi adalah:

$$\sum_{i=1}^{n} (y^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)})^2 \quad (2.12)$$

persamaan didalam tanda kurung merupakan selisih antara nilai aktual dan nilai prediksi untuk observasi ke – i. Selisih antara nilai aktual dan nilai prediksi ini diambil nilai yang terkecil, sebagai nilai score untuk model linear regression. Ordinary least squares (OLS) merupakan perhitungan nilai estimator yang paling sederhana dan yang paling banyak dipergunakan dalam mengukur score dari model linear regression. Nilai perhitungan (*estimate*) OLS ini, dipergunakan untuk menganalisa data observasi. Metode OLS ini mengambil nilai minimum dari penjumlahan kuadrat residual (selisih antara nilai prediksi dan nilai aktual).

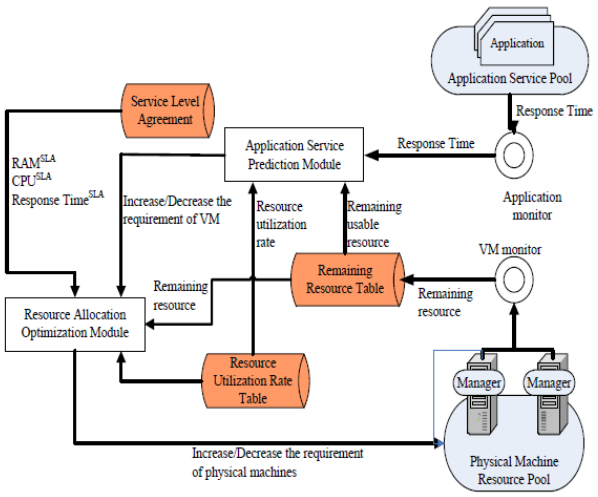
Linear regression merupakan algoritma yang paling baik (*excellent*) dan sederhana dalam mengolah data-data numerik untuk menghasilkan prediksi, dan banyak dipergunakan secara luas dalam berbagai perhitungan statistik selama beberapa dekade. Meski memiliki kekurangan karena kelinierannya. Untuk itu, untuk data dengan model yang non-linier, pendekatan terbaik (*best-fitting*) dalam mendapatkan garis lurusnya (*straight line - linearity*) dengan menggunakan selisih

kuadrat-rata-rata terkecilnya (*least mean-squared difference*). Memang garis lurusnya tentu saja tidak selalu tepat, namun model dengan pendekatan linier merupakan langkah awal untuk melangkah ke metode learning yang lebih kompleks (*building blocked for more complex learning methods*, (Torres Viñals, 2010)).

2.10.2. Model Support Vector Regression

Pendekatan dengan suatu prediksi (*prediction mechanism*) lainnya, dapat dimungkinkan dengan menggunakan Support Vector Regression (SVR). SVR dipergunakan dalam memperkirakan (*estimate*) waktu yang diperoleh (respon time) pada beberapa rentang pengukuran berikutnya, dengan sumber daya aktual yang saling di kombinasi ulang (redistributed resources) pada beberapa virtual machine yang terpasang pada sejumlah physical machine.

Gambar berikut, memperlihatkan skema penelitian yang dilakukan Huang *(Huang et al., 2013*):



Gambar 2. 5. Arsitektur Sistem Penelitian oleh Huang (Huang et al., 2013).

Gambar 2.5, memperlihatkan arsitektur sistem penelitian, yang terdiri atas Aplikasi *service resource pool* dipergunakan untuk mengelola aplikasi yang akan diuji, kemudian aplikasi monitor diperlukan untuk merekam utilisasi dari penggunaan *resource* atas aplikasi yang akan diuji. *Physical machine resource pool* dipergunakan untuk menyediakan resource, seperti CPU dan Memori, untuk host; serta dua tabel *look-up*, yaitu tabel resource tersisa (*remaining resource table*) dan tabel utilisasi resource (*resource utilization rate table*), yang dipergunakan untuk menentukan apakah resource aplikasi service tersebut perlu penambahan atau pengurangan VM. Modul Aplikasi prediksi (*application service prediction module*) dibangun menggunakan Support Vector Regression (SVR), berguna untuk memperkirakan seberapa respon (*respon time*) suatu aplikasi untuk periode berikutnya. Sedangkan *global resource allocation module* menggunakan Genetic Algorithm (GA) untuk meng-utilisasi penyebaran sumber daya (redistribute the resources), dalam arti membuat beberapa VM agar memenuhi permintaan kebutuhan user, sesuai dengan SLA yang telah disepakati. Bila belum ada perjanjian kontrak kerja, berupa SLA, maka sekaligus, SLA awal dapat dibuat dalam sistem ini.

Support Vector Regression merupakan salah satu metode *supervised learning*. Dipergunakannya SVR dalam penelitian ini untuk meng-evaluasi utilisasi resource yang dipergunakan, apakah sesuai dengan SLA yang diharapkan. Untuk mengetahui pemakaian resource alokasi virtual machine pada physical machine, digunakan persamaan berikut *(Huang et al., 2013)*:

$$\hat{u} = \sqrt{\frac{1}{m}\sum_{j=1}^{m} u_j^2 - \left(\frac{1}{m}\sum_{j=1}^{m} u_j\right)^2} \qquad (2.14)$$

$$\hat{v} = \sqrt{\frac{1}{m}\sum_{j=1}^{m} v_j^2 - \left(\frac{1}{m}\sum_{j=1}^{m} v_j\right)^2} \qquad (2.15)$$

$\hat{u}$ merupakan standar deviasi untuk setiap u_j yang merupakan banyaknya processor, dan $\hat{v}$ merupakan standar deviasi untuk setiap v_j yang merupakan banyaknya memori.

Selanjutnya diperoleh nilai yang tepat (*matching score*), c sebagai berikut:

$$c = f\left((a_p - u_q)/\hat{u}\right) f\left((b_p - v_q)/\hat{v}\right) \qquad (2.16)$$

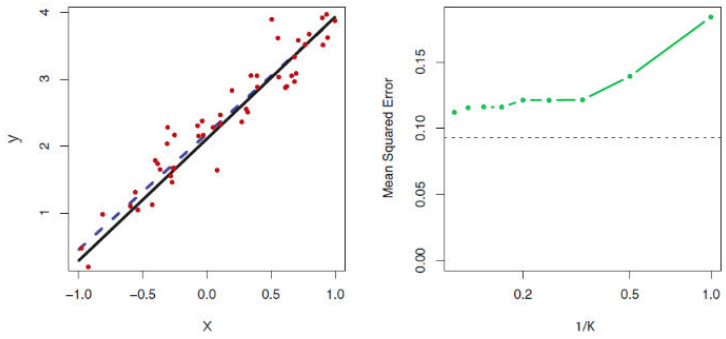
f(x) merupakan *penalty function,* a_p dan b_p masing-masing merupakan banyaknya permintaan processor dan memori atas VM dengan permintaan *p,* sedangkan u_q dan v_q merupakan processor dan memori yang disediakan oleh Physical machine sebanyak *p*.

2.10.3. Model K – Nearest Neighbors Regression

Metode KNN regression, pada data observasi untuk nilai *K* dengan prediksi pada titik x_0 , dilambangkan sebagai N_0 , dirumuskan untuk nilai prediksinya sebagai berikut (James et al., 2013):

$$\hat{f}(x_0) = \frac{1}{K}\sum_{x_i \in \mathcal{N}_0} y_i. \qquad (2.17)$$

Secara umum, nilai optimum untuk K, ditentukan dalam kurva bias-variance tradeoff. Untuk nilai K yang kecil model yang dihasilkan dapat lebih flexible, dengan bias yang kecil namun variance yang tinggi. Variance yang tinggi disebabkan bahwa model prediksi pada data yang dipilih, tergantung pada observasinya. Sebaliknya, untuk nilai K yang tinggi, akan menghasilkan model yang lebih halus dan nilai prediksinya merupakan rata-rata pada beberapa titiknya, yang tentunya perubahan antar observasi hanya berdampak kecil. Lalu bagaimana sebaiknya konfigurasi KNN untuk pendekatan data regresi? KNN dapat dipergunakan untuk data regresi secara linier (parametric) bila model linier dipilih mendekati bentuk fungsi *f* nya. Bahkan, bila hubungan antara X dan Y benar-benar linier, maka model non-parametric (non linier) sangat tidak tepat dibandingkan dengan model linear regression nya. Sebagai gambaran dapat dilihat gambar berikut



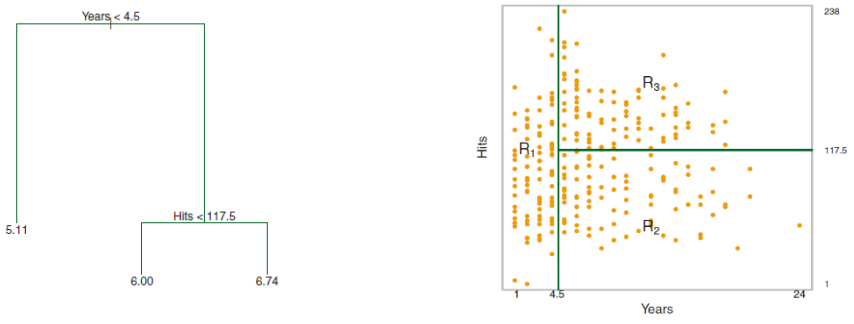
Gambar 2. 6. Korelasi antara x dan y *(James et al., 2013)*

Gambar 2.6. Sebelah kiri memperlihatkan garis biru putus-putus merupakan prediksi, sedangkan f(X) merupakan garis sesungguhnya (fact linear). Regresi linier menghasilkan prediksi yang baik untuk f(X). Sedangkan sebelah kanan memperlihatkan garis horisontal putus-putus menggambarkan MSE untuk nilai

terkecil, sedangkan garis hijau menggambarkan MSE untuk KNN sebagai fungsi 1/K (skala log). Linear regression menghasilkan MSE test yang lebih kecil dibanding KNN regression selama f(X) linier. Untuk KNN regression, hasil terbaik terjadi untuk K dengan nilai terbesar. Gambar 2.6 diatas memperlihatkan ketika hubungan X dan Y linier, maka model KNN dapat dipergunakan sebagai perbandingan terhadap model linear regression. Meski model KNN banyak dipergunakan untuk model yang non linier, namun dalam prakteknya, meski hubungan antara X dan Y walapun tidak linier, model KNN tetap dapat dipergunakan dalam memberi kesimpulan terhadap model linear regression (James et al., 2013).

2.10.4. Model Decision Tree Regression

Model decision tree dapat dipergunakan baik untuk persoalan regresi maupun klasifikasi (James et al., 2013).



Gambar 2. 7. Model Regression Tree

Gambar 2.7 sebelah kiri memperlihatkan regression tree untuk data hitters (pemukul baseball) dalam memprediksi pendapatan (salary) pemain baseball berdasarkan tahun lama bermain dalam liga dan banyaknya hits yang dibuat

dalam tahun sebelumnya. Sedangkan gambar sebelah kanan memperlihatkan pendekatan model regression tree untuk tiga variabel input R1, R2 dan R3, terbagi dalam tiga partisi untuk Y (Hits) dari data decision tree dari gambar sebelah kiri. Gambar 2.7 memperlihatkan bahwa regression tree (model Decision Tree Regression) dapat dipergunakan atas data observasinya. Regression tree yang diperlihatkan dalam gambar memperlihatkan hubungan antara Hits, Years dan Salary. Terlihat bahwa model Regression tree dapat menjadi keunggulan karena memudahkan interpretasi serta grafis yang menarik. Dalam membuat model regression tree, berikut ini algoritma modelnya (James et al., 2013):

i. gunakan splitting terus menerus (recursive binary splitting) untuk meningkatkan tree yang besar pada data trainning, lalu berhenti hanya bila setiap node terminal lebih sedikit dibanding jumlah minimum observasinya.

ii. Gunakan setiap kemungkinan subtree dari regression tree menggunakan index yang nonnegatif hingga mendapatkan tree yang besar dalam rangka mendapatkan urutan subtree yang terbaik, sebagai fungsi α .

iii. Gunakan K-fold cross-validation dalam memilih α. Selanjutnya, bagi data training kedalam K – fold. Untuk setiap k = 1, …, K:

   a. ulangi langkah 1 dan 2 secara menyeluruh kecuali untuk k training data

   b. evaluasi prediksi MSE (Mean Squared Error) sebagai fungsi dari α , buat rata-rata untuk setiap nilai α, dan ambil nilai α, untuk meminimalkan error rata-rata.

iv. Ulangi subtree dari langkah 2 untuk memilih nilai α.

2.10.5. Model Random Forest Regression

Model Random forest merupakan model lanjutan dari model decision tree. Bila model decision tree dapat dipergunakan untuk persoalan regresi maupun klasifikasi, maka model random forest pun dapat dipergunakan baik untuk persoalan regresi maupun klasifikasi. Ketika dipergunakan untuk persoalan klasifikasi, random forest menghasilkan kelas (class vote) dari setiap tree, yang selanjutnya dibuat pengelompokkan berdasarkan mayoritas (majority vote). Ketika dipergunakan untuk persoalan regresi, hasil prediksi dari setiap tree pada suatu titik target x dilakukan dengan cara membuat rata-ratanya, seperti dalam persamaan berikut (Hastie et al., 2009):

$$\hat{f}_{\mathrm{rf}}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T(x;\Theta_b) \tag{2.18}$$

yang merupakan persamaan prediksi untuk banyaknya pertumbuhan tree (B) $\{T(x;\Theta_b)\}_1^B$ sebesar sedangkan Θ_b merupakan random forest tree dengan split variabel, titik potong (cutpoints) di setiap node, dan nilai terminal-node. Adapun algoritma random forest untuk regresi maupun klasifikasi adalah

1. untuk tree dengan b = 1 hingga B:

    a) gambar sample bootstrap Z* dari N training data

    b) tambahkan random-forest tree T_b ke data bootstrap, dengan mengulangi langkah berikut untuk setiap terminal node dari tree, hingga tercapai node minimum n_{min} :

        i. Pilih *m* variabel secara acak dari variabel *p*.

        ii. Ambil variabel terbaik/split-point dalam *m*.

iii. Split node menjadi dua nodes (daughter nodes)

2. Hasilnya berupa rangkaian tree $\{T_b\}_1^B$

Selanjutnya, prediksi di suatu titik *x*, dirumuskan:

$$\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

Untuk Regresi :

$$\hat{C}_b(x)$$

Untuk Klasifikasi : Misalkan $\hat{C}_{\mathrm{rf}}^B(x) = majority\ vote\ \{\hat{C}_b(x)\}_1^B.$ merupakan class prediction dari suatu random-forest tree yang ke- *b*, maka

2.10.6. Kriteria Penentuan Model

Perhitungan dalam menentukan ukuran suatu model (*measure of fit*), menurut *Trevor Hastie* dan *Robert Tibshirani* dalam bukunya '*An Introduction to Statistical Learning*', (*James et al., 2013*), dengan menggunakan dua ukuran, yaitu: *residual standard error* (RSE) atau root mean square error (RMSE) dan R^2 statistic.

Residual standard error dirumuskan sebagai berikut

$$\text{RSE} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \quad (2.19)$$

Sedangkan $\mathbf{R^2}$ statistic mengukur hubungan linier antara X dan Y, dikenal sebagai ***correlation*****,** dirumuskan sebagai berikut

$$\frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

R^2 = (2.20)

yang memperlihatkan ***correlation*** antara x dan y, sekaligus merupakan proporsi varian (***proportion of variance***) yang memiliki nilai diantara 0 dan 1. Rumus R^2 dapat pula dituliskan sebagai berikut:

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}} \quad (2.21)$$

dengan: TSS = $\sum(y_i - \bar{y})^2$,merupakan jumlah kuadrat keseluruhan nilai (*total sum of squares*, TSS) observasi.

$\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n}$, nilai rata-rata observasi (sample means).

$\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$,merupakan jumlah kuadrat selisih (*residual sum of squares*, RSS) nilai observasi

$y_i - \hat{y}_i$, merupakan selisih antara output observasi dan prediksi, $\hat{y}$.

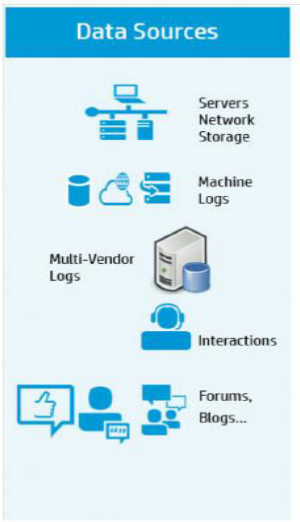
Dipertegas disini, bahwa perhitungan nilai R^2 merupakan perhitungan korelasi (hubungan) antara nilai prediksi dan nilai aktualnya. TSS mengukur total varian yang mempengaruhi Y atau dengan kata lain mengukur tingkat variasi (*amount of variability*) yang memang mempengaruhi respon sebelum proses regresi, sedang sebaliknya RSS mengukur tingkat variasi (*amount of variabiliy*) yang terjadi

setelah proses regresi. Sehingga, nilai TSS – RSS mengukur tingkat variasi (amount of variability) model regresi, dan R^2 mengukur proporsi variasi dalam menentukan nilai Y dengan menggunakan X. Nilai R^2 yang mendekati 1 berarti model regresi tepat dipergunakan. Sedang nilai R^2 yang mendekati 0 berarti model regresi tidak tepat.

2.10.7. Data Log

Machine learning dalam penelitian (thesis) ini memanfaatkan dan mengolah data berupa *log file* yang diambil dari perangkat *server*, untuk kemudian membuatnya menjadi suatu model. Adapun data *log file* diambil dari beberapa perangkat server dalam jaringan LAN (*Local Area Network*). Data dalam log inilah yang merupakan sumber utama dalam memahami bagaimana suatu mesin bekerja (*machine behavior*) (*Roy, Moitra, Das, Srinivasan, & Malhotra, 2015*). Diharapkan dengan data yang diolah menggunakan machine learning ini, prediksi atas kebutuhan *server* dapat di antisipasi dengan lebih baik. Untuk itulah proses *modeling* dengan *machine learning* ini sangat membantu selama proses disain hingga peningkatan performa yang optimal dari infrastruktur teknologi informasi.

Menurut (Roy et al., 2015), sumber data dalam pengelolaan infrastruktur TI dapat diperoleh dari log *server*. Kemudian data log dari perangkat tersebut di ambil dan dikumpulkan untuk di analisa, karena memang kinerja mesin tersimpan dalam Log tersebut dan merupakan sumber utama dalam memahami cara kerja suatu mesin (Roy et al., 2015).



Gambar 2. 8. Sumber data (Roy et al., 2015).

Gambar 2.8, memperlihatkan berbagai sumber data, seperti dari perangkat server, network maupun storage. Namun, tantangan dalam mengambil dan mengumpulkan Log ini menurut (Roy et al., 2015) adalah:

a) Banyaknya variasi produk (*Variety of Product*) dari: Servers, Storage, networking, power backup, dan sebagainya.
b) Banyaknya pembuat produk (*Multi manufacturer*): Produk (Servers, Storage, networking) dengan tipe yang sama namun dari berbagai vendor memungkinkannya dipergunakan dalam satu jaringan.
c) Tipe data (*Types of Machine Data*): setiap perangkat (mesin) memiliki style tersendiri dalam menyediakan datanya. Ada yang berbentuk *plain text*, ada yang ter-*enkripsi*, atau berbentuk *file biner* maupun berbagai tipe struktur data lainnya.

Tantangan ini dalam arti, harus mampu dijawab dengan membuat kombinasi berbagai format log tersebut menjadi satu format yang layak dan dapat dibaca sehingga dapat dimanfaatkan dalam pembuatan suatu algoritma dalam memahami dan pembuatan model yang menggambarkan kinerja/performa infrastruktur TI

baik secara keilmuan maupun mampu diterapkan dalam dunia nyata. Sehingga menjadi jelas bahwa dengan memanfaatkan data pengelolaan sumber daya eksisting, dapatlah dibuat suatu rencana untuk pengelolaan sumber daya di masa mendatang. Sehingga tata kelola sumber daya berbasis data eksisting memang sebaiknya digunakan untuk membuat prediksi, membantu penyusunan rencana pengelolaan sumber daya infrastruktur TI dimasa mendatang dengan lebih baik.

2.10.8. Pemilihan Variabel (Feature Selection)

Pemilihan variabel atas data log, merupakan langkah penting dalam pembuatan model matematis aplikasi prediksi kinerja infrastruktur TI. Tabel 2.2 memperlihatkan pemilihan variabel input dan output (dari data log) menurut beberapa penelitian. Tabel memperlihatkan penelitian yang memilih variabel yang menjadi input dan output serta metode yang dipilih. Peneliti pertama, Torres Viñals, J. (2010), memilih variabel input berupa *Throughput, Response Time, System Load, Disk Usage* dan *Memory.* Sedangkan variabel output berupa waktu (*Time_until_Fau*lt). Adapun metode yang dipilih adalah Linear Reggression, Decission Tree dan Bayesian Network.

Tabel 2. 2. Tabel pemilihan variabel input dan output

	Peneliti	Variabel		Metode
		Input	Output (Prediction)	
1	Torres Viñals, J. (2010). Resource management on Cloud systems with Machine Learning.	Throughput,	Time_until_Fault	Linier Reggression
		Response Time		Decission Tree
		System Load		Bayesian Network
		Disk Usage		
		Memory		
2	Huang, C.-J., Wang, Y.-W., Guan, C.-T., Chen, H.-M., & Jian, J.-J. (2013). Applications of Machine Learning to Resource Management in Cloud Computing. *International Journal of Modeling and Optimization*, *3*(2), 148–152. https://doi.org/10.7763/IJMO.2013.V3.256	System Resource:	Response Time	Support Vector Regression (SVR)
		CPU, Memory	Resource Redistributed	Genetic Algorithm (GA)
3	Roy, C., Moitra, S., Das, M., Srinivasan, S., & Malhotra, R. (2015). IT Infrastructure Downtime Preemption using Hybrid Machine Learning and NLP, *6*, 39–44. https://doi.org/10.15439/2015F400	Log Server	Real Time Faults	Support Vector Machine
		Monitoring	and downtime	Random Forest
4	Gao, J., & Jamidar, R. (2014). Machine Learning Applications for Data Center Optimization. Google White Paper, 1–13.	IT Load, Chillers,	Energy Efficiency	Neural Network
		Cooling		

Peneliti kedua, Huang, C.-J., Wang, Y.-W., Guan, C.-T., Chen, H.-M., & Jian, J.-J. (2013), memilih variabel input berupa System Resource: CPU dan Memory. Sedangkan variabel output berupa *response time* dan *resource redistributed*. Adapun metode yang dipilih adalah Support Vector Regression (SVR) dan Genetic Algorithm (GA).

Peneliti ketiga, Roy, C., Moitra, S., Das, M., Srinivasan, S., & Malhotra, R. (2015), memilih variabel input berupa System Resource monitor dalam log server dan variabel output berupa waktu ketika terjadi gangguan (*Real Time Faults* and *downtime*). Adapun metode yang dipilih adalah Support Vector Machine dan Random Forest.

Peneliti keempat, Gao, J., & Jamidar, R. (2014), memilih variabel input berupa IT Load, dan sistem pendinginan (c*hillers, cooling*) dan variabel output berupa efisiensi energi. Adapun metode yang dipilih adalah Neural Network.

Penelitian dalam thesis ini menggunakan variabel input, berupa cpu usage, disk usage dan memory usage, sedangkan variabel output berupa network performance yang keseluruhannya diambil dari log file system resource. Adapun metode yang dipergunakan dalam penelitian ini, menggunakan beberapa model, seperti: Linear Regression, kNN, Support Vector Regression (SVR), Decision Tree dan Random Forest.