Analisis Prediktif dengan Machine Learning Classifier untuk Prediksi Penyakit Stroke

Git Hub Link : https://github.com/axeltanjung/stroke_prediction

1. Background

Menurut WHO, stroke adalah penyebab ke dua yang menyebabkan kematian global, yang bertanggung jawab terhadap sekitar 11% total kematian di dunia. Dampak negatif dari stroke tersebut meningkatkan kebutuhan untuk inovasi dasi sisi diagnosis dan managemen terhadap penyakit stroke. Peluang untuk meningkatkan ketepatan hasil diagnosis terhadap potensi seseorang akan menderita penyakit stroke terbuka dengan menggunakan analisis terhadap data medis pasien terdahulu.

Report ini berisi analisis terhadap hubungan antara faktor faktor yang dapat mempengaruhi kemungkinan seseorang menderita penyakit stroke. Input permasalahan ini adalah beberapa fitur seperti gender, umur, Riwayat hipertensi dan serangan jantung, status pernikahan, tipe pekerja, tipe residental, rata-rata tingkat kadar glukosa darah, bmi, dan status perokok. Kemudian kita gunakan beberapa model prediksi Classification yaitu KNN, Logistic Regression, Decision Tree, dan Random Forest. Output akhir dari penelitian ini adalah analisis terhadap fitur apa saja yang berkolerasi terhadap meningkatkan penyakit stroke dan model prediktif yang dapat memprediksi potensi seseorang menderita penyakit stroke sehingga diharapkan dengan adanya penelitian ini dapat membantu tenaga medis untuk menentukan langkah lanjutan dalam treatment terhadap pasien.

2. Related Works

Penulis menambahkan beberapa related work untuk menambahkan point of view baru terhadap konsep Prediksi

- Machine Learning Performance Analysis to Predict Stroke Based on Imbalanced Medical Dataset – Yuru Ling

Berdasarkan jurnal tersebut, walaupun tidak terdapat bentuk distribusi yang berbeda siginifikan antara tiga fitur numerik (Age, Avg Glucose, dan BMI), rata-rata nilai dari ketiga fitur itu lebih tinggi saat seseorang menderita stroke. Fitur hypertension dan heart disease potensial berkaitan juga dengan stroke. Namun berdasarkan analisis DCT bahwa fitur heart dieses dan ever married merupakan fitur yang tidak berdampak signifikan. Namun fitur tersebut tetap digunakan karena berdasarkan medical judgement bahwa fitur tersebut berpengaruh. Tidak terdapat fitur yang bersifat kolinear secara ekstrim. Untuk preprocessing data, tidak terdapat perbedaan signifikan dalam penggunaan label encoding dan one hot encoding.

Penggunaan SMOTE sangat baik dalam handling kelas yang imbalance dibanding metode lain. Tetapi SMOTE memiliki efek negative dalam meningkatkan overlapping & noises

tambahan terutama dengan *high dimensional data.* Model terbaik yang didapatkan tanpa SMOTE adalah QDA. Sedangkan dengan menggunakan SMOTE didapatkan model SGS, DCT, dan LR yang memiliki performa baik dalam f1-score dan nilai AUC. Sedangkan penggunaan voting terbobot tidak direkomendasikan karena implementasi terhadap dataset yang besar tidak focus terhadap outlier. Imputasi yang lebih baik dapat menggunakan regresi dibandingkan simple imputation.

3. Dataset & Features

I. Deskipsi Features

a. **id**: unique identifier
b. **gender**: "Male", "Female" atau "Other"
c. **age**: Umur dari pasien
d. **hypertension**:
   "No" jika pasien tidak menderita hipertensi
   Normal dengan tekanan darah sistolik < 120 mmHg dan diastolic < 80 mmHg
   Pre-hipertensi dengan tekanan darah sistolik 120-139 dan diastolic 80-89)
   "Yes" jika pasien menderita hipertensi
   Hipertensi derajat 1 dengan tekanan darah sistolik 140-159 mmHg dan diastolic 90-99 mmHg
   Hipertensi derajat 2 dengan tekanan darah sistolik >160 mmHg dan diastolic >100 mmHg
e. **heart_disease**: "No" jika pasien tidak pernah menderita penyakit jantung, "Yes" jika pasien pernah menderita penyakit jantung
f. **ever_married**: "No" atau "Yes"
g. **work_type**: "children", "Govt_jov", "Never_worked", "Private" atau "Self-employed"
h. **Residence_type**: "Rural" atau "Urban"
i. **avg_glucose_level**: rata-rata glukosa dalam darah (mg/dL)
j. **bmi**: body mass index
   Underweight
   Batas Normal
   Overweight
   Pre-obese
   Obese I
   Obese II
k. **smoking_status**: "formerly smoked", "never smoked", "smokes" or "Unknown"*
l. **stroke**: 1 if the patient had a stroke or 0 if not
   *Note: "Unknown" in smoking_status means that the information is unavailable for this patient

II. Umum

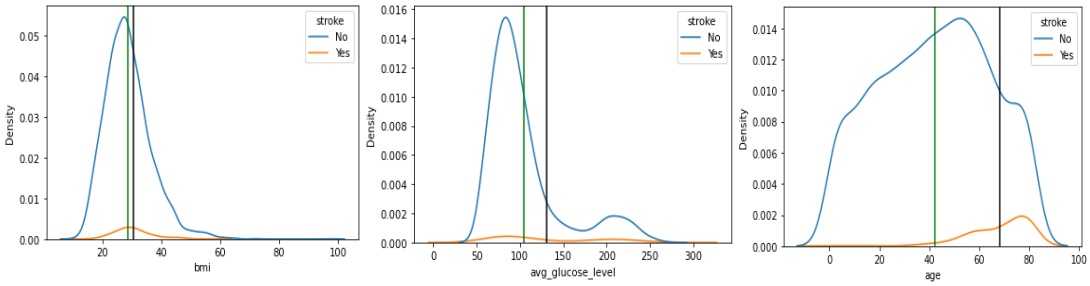
Secara umum, terdapat 12 features / kolom dalam dataset tersebut dan 5.256 data observasi. Setelah dilakukan pengecekan terdapat 146 data observasi terduplikat sehingga dilakukan dropping dan menghasilkan 5.110 data observasi. Terdapat 3.832 data training, 5 k Fold pada data validation, dan 1278 data test. Selanjutnya dilakukan *data preprocessing* dengan melakukan *split input* dan *output* yaitu pemisahkan kolom

"stroke" menjadi variable y dan lainnya menjadi kolom x. Setiap tahapan tersebut dibuat dalam bentuk fungsi agar dapat dilakukan secara berulang.

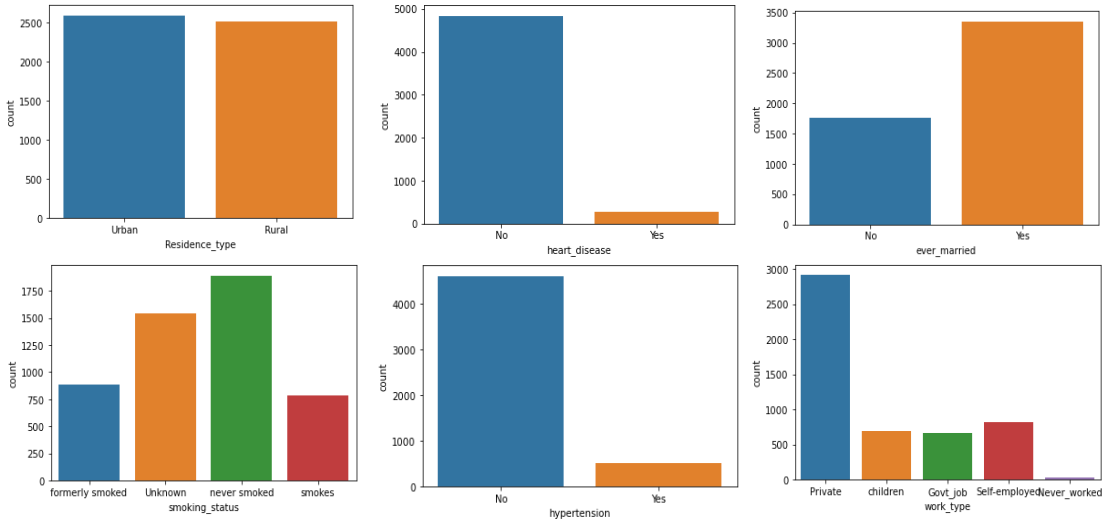
Selanjutnya dilakukan pengecekan terhadap *unique value* pada setiap fitur yaitu gender : 3, age : 104, hypertension : 2, heart_disease : 2, ever_married : 2, work_type : 5, Residence_type : 2, avg_glucose_level : 3979, bmi : 418, smoking_status : 4. Dilakukan drop pada fitur id karena merupakan unique value yang tidak mewakili. Fitur gender dikompresi menjadi 2 nilai unik dimana gender *others* dimasukkan ke kategori *female* sesuai dengan modus data. Selanjutnya dilakukan train-test split agat tidak terdapat overfit data training dimana test data akan menjadi future data. Test size diberikan sebesar 0.25 dengan metode stratify karena datanya imbalance.

III. Exploratory Data Analysis

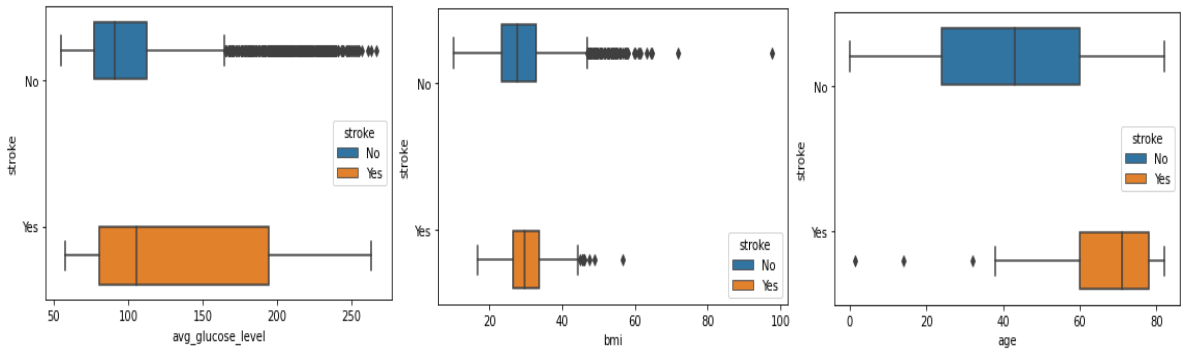
Setelah dilakukan splitting data, selanjutnya dilakukan Exploratory Data Analysis untuk dataset. Dilakukan pengecekan skewness pada fitur numerik yaitu *age (-0.129150), avg_glucose_level (1.583383),* dan *bmi (1.018919)*. Jumlah dari penderita stroke adalah 187 orang dibanding non-penderita sebesar 3645 orang (dari total data training) sehingga dikategorikan sebagai data imbalance. Gender yang dominan adalah Female (2.266) orang atau 59.13% dari total sample dibanding Male (1.566) orang. Plotting distribus dilakukan terhadap variable age, bmi, dan avg_glucose_level untuk melihat *probability density function* dan dibandingkan terhadap kategori *stroke* dan *non-stroke*



Sedangkan untuk fitur categorical lainnya seperti hipertensi, penyakit jantung, status pernikahan, tipe pekerjaan, tipe residental, status perokok, dan gender dilakukan plotting pada figure berikut.



Selanjutnya dilakukan pengecekan terhadap outliers untuk variable numerik dan didapatkan boxplot sebagai berikut



Didapatkan beberapa outliers yang terdapat pada avg_glucose_level untuk penderita non_stroke, dimana kecenderungan penderita stroke memiliki avg_glucose_level yang lebih tinggi dibanding non-penderita. Sedangkan untuk nilai bmi secara *pdf* dan boxplot memiliki mean dan median yang berdekatan. Variable *age* merepresentasikan mean dan median yang berbeda signifikan antara penderita stoke dan non penderita. Dilakukan hipotesting variable numerik untuk mengecek kesamaan mean penderita stroke dan non-penderita. Didapatkan hasil hipotesis testing yaitu mean dari avg_glucose_level dan age signifikan dibanding stroke dan non stroke dibandingan dengan bmi yang memiliki mean yang tidak cukup siginifikan.

IV. Data Imputation (Preprocessing)

Pada fase ini, fitur dibagi menjadi numerical & categorical. Terdapat 3 fitur yang memiliki NaN value yaitu work_type, Residence_type, dan bmi. Untuk bmi yang merupakan kategori numerical, dilakukan imputation menggunakan *SimpleImputer* dari *Sklearn* dengan strategi imputasi *median*. Sedangkan untuk fitur work_type, dan Residence_type, dilakukan input kategori "Kosong" untuk NaN value.

Setelah tidak terdapat nilai yang kosong, dilakukan preprocessing untuk categorical variables menggunakan One Hot Encoding dan Label Encoding. OHE diaplikasikan untuk fitur yang memiliki lebih dari 2 unique value (work_type, smoking_status), sedangkan LE diimplementasikan pada fitur yang memiliki 2 unique value (gender, hypertension, heart_disease, ever_married, Residence_type.

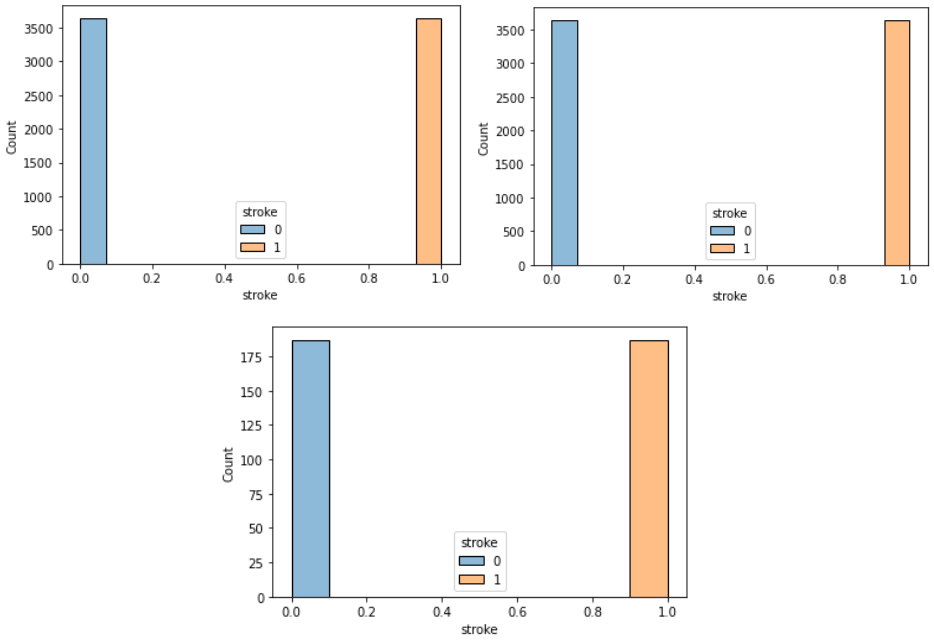
Setelahnya dilakukan penggabungan Kembali menggunakan concat. Setelah dipastikan semua fitur sudah berubah menjadi angka, dilakukan standarisasi variable dikarenakan metode yang digunakan untuk Machine Learning merupakan *distance based* yang tidak robust terhadap range data yang lebar seperti KNN dan Logistic Regression untuk Ridge dan Lasso. Standarisasi dilakukan menggunakan *StandardScaler* dari *SKLearn.* Berikut merupakan dataset hasil dari standarisasi yang telah dilakukan

	age	avg_glucose_level	bmi	gender	hypertension	heart_disease	ever_married	Residence_type	work_type_Govt_job	work_type_KOSONG
5050	-0.238216	0.081395	1.662635	-0.831316	3.063466	-0.237127	0.722387	-1.005826	-0.391819	-0.016156
1530	-0.766771	0.241311	-1.312734	-0.831316	-0.326428	-0.237127	-1.384300	0.977872	-0.391819	-0.016156
623	-0.282262	-0.692716	0.096651	-0.831316	-0.326428	-0.237127	0.722387	0.977872	-0.391819	-0.016156
4370	-0.678679	-0.197684	0.031402	-0.831316	-0.326428	-0.237127	-1.384300	-1.005826	-0.391819	-0.016156
5069	-0.810818	-0.995495	-0.555842	-0.831316	-0.326428	-0.237127	0.722387	0.977872	-0.391819	-0.016156

work_type_Never_worked	work_type_Private	work_type_Self-employed	work_type_children	smoking_status_Unknown	smoking_status_formerly smoked	smoking_status_never smoked	smoking_status_smokes
-0.068698	0.867278	-0.435537	-0.3896	-0.650751	-0.460733	1.288124	-0.422317
-0.068698	0.867278	-0.435537	-0.3896	-0.650751	2.170456	-0.776323	-0.422317
-0.068698	0.867278	-0.435537	-0.3896	1.536686	-0.460733	-0.776323	-0.422317
-0.068698	0.867278	-0.435537	-0.3896	1.536686	-0.460733	-0.776323	-0.422317
-0.068698	0.867278	-0.435537	-0.3896	-0.650751	-0.460733	1.288124	-0.422317

V. Sampling

Data penderita *stroke* dan *non-stroke* merupakan data yang imbalance, sehingga untuk eksperimentasi performa model dilakukan konfigurasi terhadap sample data menggunakan Random Under Sampling (374 observasi), Random Over Sampling (7290 observasi), dan Oversampling (SMOTE) (7290 observasi).



4. Metode

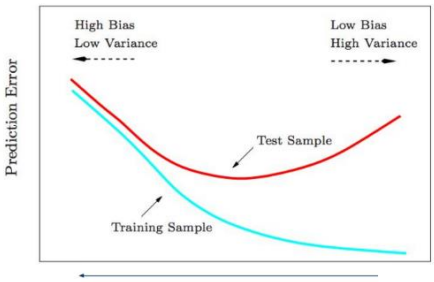
I. K-Nearest Neighbor (KNN)

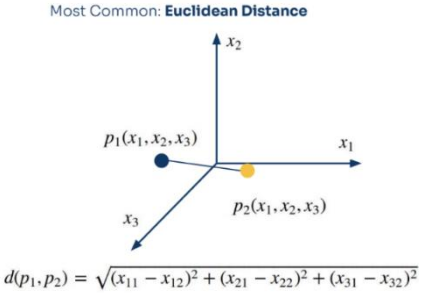
Algoritma ini mengestimasi nilai berbasis titik menggunakan tetangga terdekat (nearest neighbor). Sejumlah k observasi yang memiliki jarak terdekat dengan titik, misalkan, X. Metode ini bisa digunakan untuk melakukan prediksi secara Regresi dan Klasifikasi. Untuk kasus Regresi digunakan Mean/Average dari nilai target, sedangkan Klasifikasi menggunakan majority vote

Regression	Classification
Mean/Average: $f(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$	Majority Vote: $f(x) = \frac{1}{n}\sum_{i=1}^{n} I(y_i = 1)$

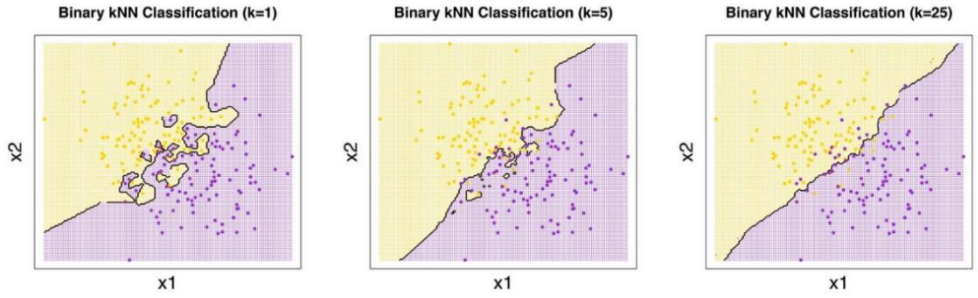
Tahapan yang dilakukan pada KNN adalah pertama pilih titik observasi dimana telah dilakukan ploting pada kartesian terhadap dua buah fitur. Selanjutnya tentukan nilai *k* atau jumlah tetangga terdekat dari titik observasi, sebagai contoh diambil nilai k

sebesar 3. Selanjutnya hitung nilai average/mean (regression) dan lakukan majority vote (classification) untuk masing-masing titik. Tetangga terdekat dihitung berdasarkan jarak terdekat dengan titik observasi. Menghitung jarak dari titik observasi dapat dilakukan menggunakan banyak cara, antara lain Euclidian & Manhattan sesuai dengan persamaan berikut.





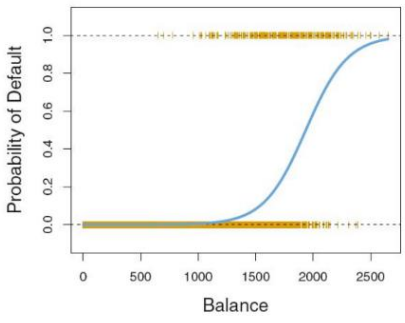
Nilai dari k terbaik tergantung pada data. Cross validation dapat digunakan untuk membandingkan k. Nilai K yang semakin kecil meningkatkan potensi overfit / menurunkan bias pada data sehingga perlu dengan menambah kompleksitas dari model. Sedangkan nilai K yang besar akan menurunkan sensitifitas terhadap noise data, dan membuat model lebih sederhana tetapi meningkatkan bias pada model yang disebut dengan *Bias Variance Tradeoff*



Model K observasi terdekat pada titik observasi x mungkin menjadi jauh jika memiliki dimensi p yang besar. Untuk mengatasi hal tersebut dapat dilakukan feature selection, menggunakan distance function yang lebih complex, dan menggunakan parametric model (Linear Regression / Logistic Regression)

II. Logistic Regression

Algoritma ini merupakan turunan dari algoritma linear regression. Apabila menggunakan linear regression, kita dapat memodelkan nilai yang besar lebih dari angka 0 dan 1. Dibanding menggunakan nilai tersebut, kita dapat membatasi hasil dari output linear tersebut menggunakan upper-bounded to 1 dan lower bounded to 0.



Tahapan yang dilakukan pertama asumsi bahwa targe output kontinu. Selanjutnya kita dapat melakukan estimasi liniear regresi dengan fungsi berikut:

$$\hat{y} = f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$
$$= w_0 + \mathbf{w}^T \mathbf{x}$$

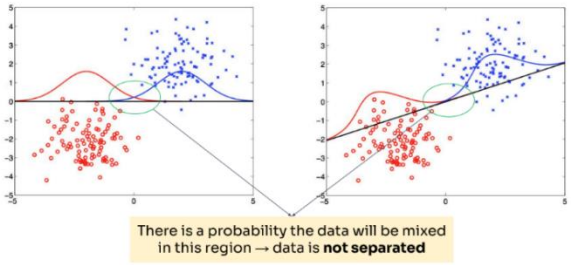
Selanjutnya, kita dapat mencari nilai w yang minimal yang meminimalkan fungsi berikut

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Kita dapat menggunakan hasil dari fungsi regresi untuk klasifikasi data baru dengan konfigurasi berikut:

- Label = 1 if f(x, w) > 0.5
- Label = 0 otherwise

Bagaimana cara untuk mendapatkan decision boundary terbaik? Dengan fungsi linear regresi yang disebutkan diatas, proyeksikan tiap data pada garis w. Selanjutnya buat pdf untuk tiap kelas. Terdapat kemungkinan data tergabung.



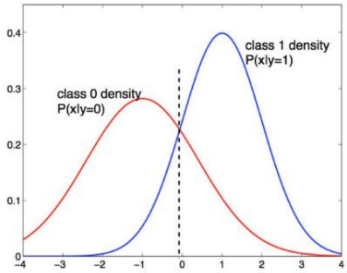
Kita perlu mencari garis yang dapat memaksimalkan proyeksi separasi pada tiap kelas. Optimal decision bergantung pada probalilitas posterior kelas dengan rumusan sebagai berikut

$$P(y \mid \mathbf{x})$$

- y = 1 if

$$\log \frac{P(y=1 \mid \mathbf{x})}{P(y=0 \mid \mathbf{x})} > 0$$

- y = 0 otherwise



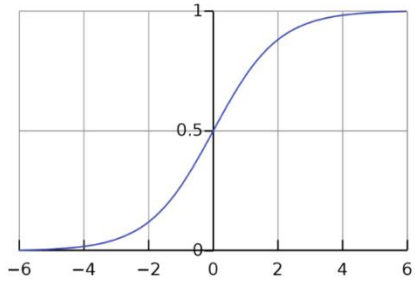
Kita tidak mengetahui nilai dari P(x|y). Kita dapat melakukan parameter dengan fungsi berikut

$$\log \frac{P(y=1 \mid \mathbf{x})}{P(y=0 \mid \mathbf{x})} = f(\mathbf{x}; \mathbf{w}) = w_0 + \mathbf{x}^T \mathbf{w} \qquad P(y=1 \mid \mathbf{x}) = g(w_0 + \mathbf{x}^T \mathbf{w})$$

Dimana

$$g(x) = \frac{1}{1 + e^{-x}}$$

g(x) adalah fungsi yang mengubah fungsi linier menjadi probabilitas



Kita dapat menggunakan fungsi liniear regression yang lama dan memasukkannya ke fungsi sigmoid

$$\beta_0 + \beta_1 x_1 \qquad p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1)}}$$

Decision boundary Ketika probabilitas = 0.5

$$\frac{P(y = 1 \mid \mathbf{x})}{P(y = 0 \mid \mathbf{x})} = \frac{0.5}{0.5} = 1.0$$

$$\log \frac{P(y = 1 \mid \mathbf{x})}{P(y = 0 \mid \mathbf{x})} = \log 1.0 = 0.0$$

$$w_0 + \mathbf{w}^T \mathbf{x} = 0.0$$

Untuk model logistic ini tidak terdapat solusi tertutup seperti OLS. Dapat dilakukan Gradient Descent untuk mendapatkan solusi dari persamaan tersebut. Dapat dilakukan regularisasi menggunakan L1 norm dan L2 norm

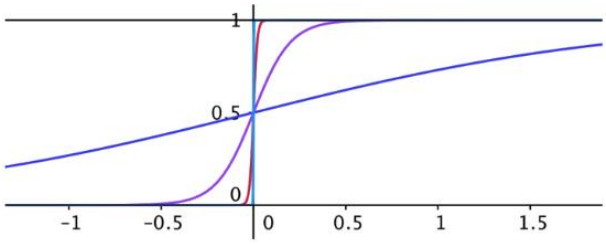
Berikut merupakan logistic regression yang telah dilakukan regularisasi seperti linear regression:

Logistic Regression ⟶ $\text{Log-loss} = \sum_{j=1}^{n} \left(y_j \left(\beta_0 + \sum_{i=1}^{p} \beta_i x_i \right) - \log \left(1 + \exp \left(\beta_0 + \sum_{i=1}^{p} \beta_i x_i \right) \right) \right)$

Ridge Regression ⟶ $= \text{Log-loss} + \lambda \sum_{j=1}^{p} \beta_j^2$

Lasso Regression ⟶ $= \text{Log-loss} + \lambda \sum_{j=1}^{p} |\beta_j|$

where $||\beta||_1 = \sum |\beta_j|$



$$\mathbf{f(x)} = \frac{1}{1 + e^{-x}}$$

$$\mathbf{p(x)} = \frac{1}{1 + e^{-10x}}$$

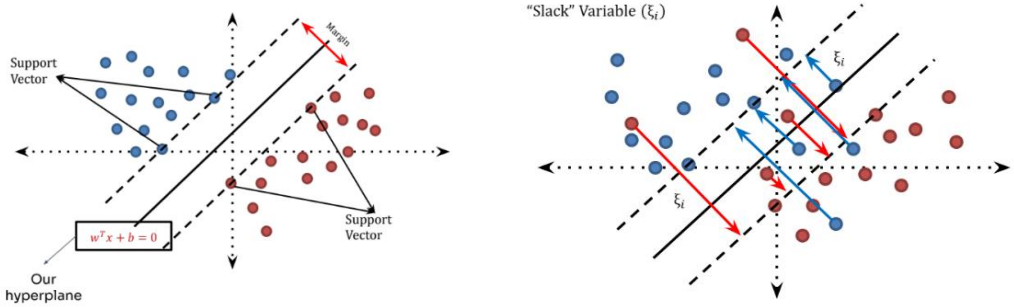
$$\mathbf{q(x)} = \frac{1}{1 + e^{-100x}}$$

$$\mathbf{r(x)} = \frac{1}{1 + e^{-1000x}}$$

Regularisasi dapat membuat weight koefisien pada linear regression berkurang mendekati nol

III. Support Vector Machine

Algoritma ini melakukan separasi data menggunakan hyperplane. Untuk data yang bersifat satu dimensi, hyperplane yang terbentuk berupa titik. Untuk data yang berbentuk 2 dimensi akan berbentuk garis. Untuk data yang berbentuk >3 dimensi akan berbentuk hyperplane yang tidak dapat divisualkan. Kita dapat membuat hyperplane secara tidak terbatas dan menentukan hyperplane yang memaksimalkan margin antara class.



Karena data tidak terseparasi 100%, maka dapat digunakan soft margin classifier yang digambarkan oleh nilai slack ξ. Nilai 1 > ξ > 0 menandakan bahwa titik berada pada margin dan berada pada posisi yang benar pada hyperplane. Nilai 1 < ξ menandakan terdapat misklasifikasi pada poin.

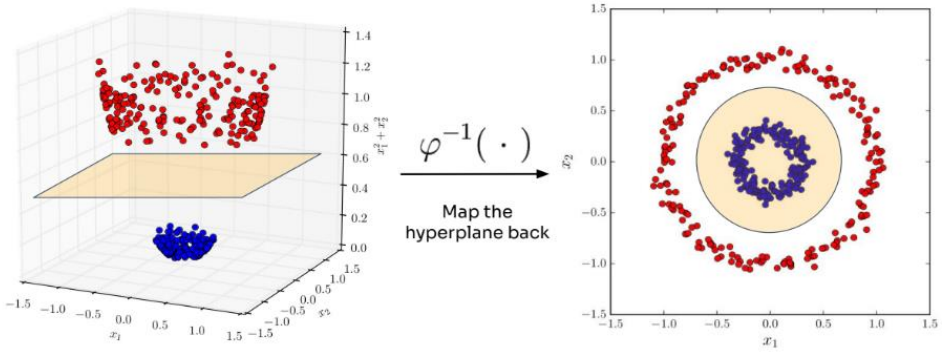
$$\min_{w,b,\xi} = \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$

"cost" parameter → similar to regularization

$$\text{s.t.} \quad y_i\left(w^T x_i + b\right) \geq 1 - \xi_i, \quad \xi \geq 0, \quad \text{for } i = 1, 2, \cdots, n$$

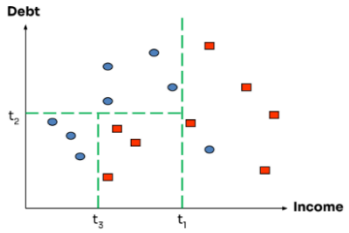
Nilai C merupakan hyperparameter yang menentukan margin, dimana nilai C yang kecil memiliki margin yang besar sedangkan nilai C yang besar menandakan nilai margin yang kecil.

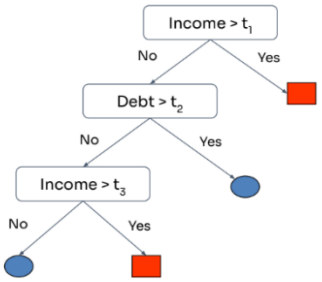
Apabila liniear boundary tidak dapat secara efektif untuk melakukan pemisahan pada data, dapat ditambahkan variable baru berupa kernel function. Contoh dari kernel function adalah linear function, D-th degree polynomial, dan Radial Basis Function (RBF)



IV. Decision Tree

Algoritma Decision Tree membagi dataset menjadi beberapa region (berbentuk kotak). Pilih fitur & threshold yang dapat membagi target dengan baik. Ulangi Langkah tersebut hingga max_depth yang dikehendaki yang akhirnya akan menciptakan boundary untuk mengelompokkan data berdasarkan region yang telah dibuat tadi.





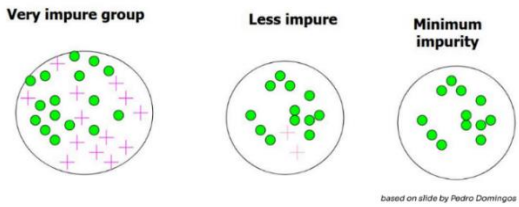
Terdapat beberapa komponen dari Decision Tree yaitu Decision / Internal Nodes, Leaf Nodes, Branch, dan Root Nodes. Bentuk dari Decision Tree seperti pohon yang terbalik dimana Root berada di atas. Decision Node merupakan uji nilai dari salah satu atribut atau fitur. Kumpulan edges/branches dari node yaitu label dengan nilai atribut/ fitur yang mungkin. Dan leaf node merupakan output value.

Tahapan untuk melakukan split adalah menggunakan perhitungan MSE (Mean Squared Error) disetiap angka pada feature space. Tree dibentuk secara rekursif, satu branch tiap waktu. Split ditentukan pada kondisi yang paling besar menurunkan cost function J. Algoritma dihentikan Ketika stopping criterion tercapai

Regression	Classification
Output: • $f(\mathbf{x})$: $\mathbb{R}^p \rightarrow \mathbb{R}$	Output: • Binary Classification: $\Omega = \{-1,1\}$ • Multi-class Classification: $\Omega = \{1, 2, 3, \dots, C\}$ • $f(\mathbf{x})$: $\mathbb{R}^p \rightarrow \Omega$

Untuk kasus klasifikasi, terdapat masalah untuk memilih atribut terbaik untuk memisahkan data. Opsi yang dilakukan dapat secara random, least values (possible value terkecil), most value (possible value terbesar), dan max gain (information gain terbesar / mempercepat ukuran tree).

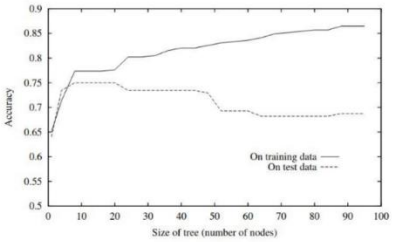
Konsep entropy dapat menggambarkan tingkat impurity dari suatu grup. Untuk kelas yang sangat impure memiliki keacakan yang tinggi sedangkan yang minimum impure memiliki kecenderungan kelas yang homogen. Nilai entropy didapatkan dengan persamaan berikut:



$$H(x) = -\sum_{i=1}^{n} P(x=i)\log_2 P(x=i)$$

P(x=i) adalah probability dari kelas-i. Dihitung sebagai proporsi kelas-i pada set.

Set yang baik untuk model belajar adalah set yang memiliki impurity yang tinggi. Selain dengan entropy, dapat digunakan gini index & classification error untuk menghitung error. Selain itu terdapat opsi lain untuk melakukan pembagian. Atribut yang baik akan membagi data menjadi subset yang ideal yaitu semua positif atau semua negatif. Selanjutnya dilakukan pencarian atribut yang paling menurunkan entropi diketahui atribut sebelumnya yang disebut dengan information gain = entropy(parent) – [average entropy(children)]. Penambahan branch/tree akan meningkatkan akurasi prediksi. Semakin banyak tree maka akurasi prediksi yang bisa didapat mencapai 100% pada data training. Namun pada data test akan turun untuk nilai akurasinya.



$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m}) + \alpha |T|$$

Decision tree sangat rentan terhadap overfitting. Tanpa stop critrerion selama proses pembuatan tree maka dapat didapatkan akurasi 100% pada data training. Solusinya dapat dilakukan tree pruning. Tahapan yang dapat dilakukan adalah membuat tree super complex, selanjutnya dilakukan pemilihan tuning parameter alpha (non-negative float). Untuk setiap nilai alpha, akan ada subset T sedemikian hingga cost function bernilai minimum. Pruning dari performa validation set- prune tree untuk memaksimalkan perfoma validation set.

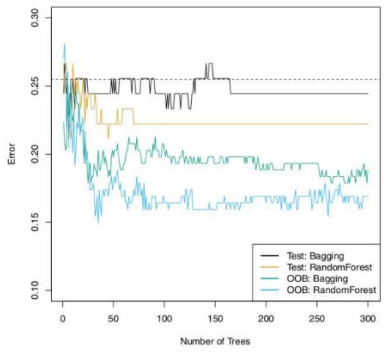
Keunggulan dari model decision tree adalah dapat memodelkan high model dimensional. Selain itu model tree mudah untuk diinterpetasikan. Tetapi model ini memiliki kekurangan yaitu terdapat kesulitan untuk memodelkan bentuk liniear. Selain itu, model ini memiliki kecenderungan memiliki variansi model yang tinggi sehingga meningkatkan potensi error dan menurunkan intrepretabilitas data. Model ini akan baik digunakan untuk model gabungan seperti boosting dan bagging. Model ini termasuk model yang tidak stabil karena merubah sedikit data dapat mengakibatkan perbedaan yang signifikan pada partisi data

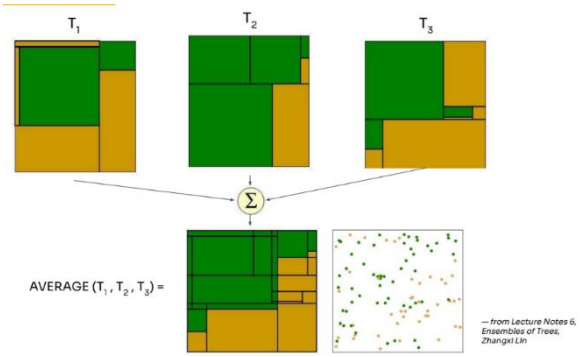
V. Random Forest (Model Ensamble)

Model ini berkaitan dengan model ensamble dimana konsep yang diberikan adalah dengan membuat beberapa model dengan erronya masing-masing. Menggunakan classifier yang kurang berkolerasi akan menurunkan nilai dari variance dan mengimprove performa model. Model tersebut didapatkan dari data hasil bootstrapping, yaitu metode resampling dengan pengembalian.

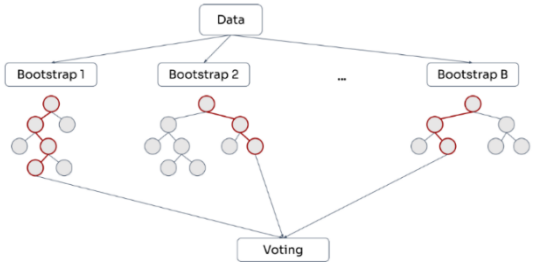
Pada konsep Decision Tree, Tree tersebut dapat menjadi tidak stabil dan memiliki variance yang tinggi. Untuk mereduksi nilai dari variance, dibuat B-Bootstraped dataset dan fit tiap pohon untuk tiap boothstapped dataset sehingga didapatkan B-Tree Model. Untuk model classification didapatkan dari majority vote.

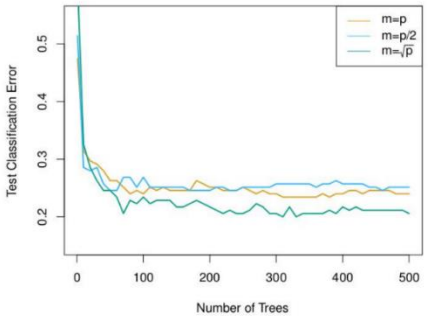
Jumlah dari Tree B bukan merupakan variable yang kritikal. Menggunakan nilai B yang tinggi tidak membuat menjadi overfit. Praktikalnya menggunakan nilai B yang besar dapat menurunkan nilai error





Untuk melakukan reduksi terhadap error, dapat dilakukan randomisasi fitur pada tree





5. Experiment/Result/Discussion

Karena nilai dari target imbalance sebesar 95%, maka metrics dari accuracy kurang dapat digunakan sebagai metrics utama yang digunakan untuk menentukan performa model terbaik. *Business case* yang terjadi pada kasus klasifikasi penderita stroke ini lebih cocok menggunakan metrics yang menurunkan nilai dari False Negative (FN).
Apabila seseorang diprediksi oleh model tidak stroke tetapi pada kenyataanya menderita stroke, maka akan menjadi lebih berbahaya karena treatment preventif terhadap observant tersebut sudah tidak dapat dilakukan. Selain itu akan lebih berbahaya juga karena tindakan medis harus dilakukan untuk melakukan penyembuhan.
Terdapat beberapa metrics yang dapat digunakan untuk model klasifikasi diantaranya

1. Accuracy : (Total Prediksi Tepat) / (Total Data Observant)
2. Precision : True Positive / (True Positive + False Positive)
3. Recall : True Positive / (True Positive + False Negative)
4. F1 Score : Harmonik mean antara precision dan recall
5. ROC (Receiver Operating Characteristics Curve) : Ploting dari TPR (True Positive Rate) dan FPR (False Positive Rate)
6. AUC (Area Under Curve) : Semakin mendekati 1 semakin baik

Dari penjelasan diatas, diperlukan nilai Recall yang baik untuk mendapatkan performa model yang baik. Selanjutnya untuk tahap eksperimentasi dilakukan fitting vanilla model (model default) terhadap X_train untuk seluruh model yang dibahas diatas. Fitting model

dilakukan juga untuk seluruh dataset yang dilakukan Random Under Sampling, Random Over Sampling, dan SMOTE. Sehingga didapatkan model sebagai berikut:

No	Model	Sampling Metode
1	K-Nearest Neighbor	Normal Sampling
2	Logistic Regression	Normal Sampling
3	Support Vector Machine	Normal Sampling
4	Decision Tree	Normal Sampling
5	Random Forest	Normal Sampling
6	K-Nearest Neighbor	Random Under Sampling
7	Logistic Regression	Random Under Sampling
8	Support Vector Machine	Random Under Sampling
9	Decision Tree	Random Under Sampling
10	Random Forest	Random Under Sampling
11	K-Nearest Neighbor	Random Over Sampling
12	Logistic Regression	Random Over Sampling
13	Support Vector Machine	Random Over Sampling
14	Decision Tree	Random Over Sampling
15	Random Forest	Random Over Sampling
16	K-Nearest Neighbor	Over Sampling (SMOTE)
17	Logistic Regression	Over Sampling (SMOTE)
18	Support Vector Machine	Over Sampling (SMOTE)
19	Decision Tree	Over Sampling (SMOTE)
20	Random Forest	Over Sampling (SMOTE)

Selain itu, dilakukan proses *Cross Validation* untuk menentukan parameter terbaik yang dapat digunakan oleh tiap-tiap model untuk meningkatkan performa dari model. Digunakan nilai k dari Grid Search CV sebesar 5 dengan artian data training tersebut di split menjadi 5 kali dan dilakukan Cross Validasi untuk tiap tiap segmen data.

Untuk model Logistic Regression, nilai dari penalty model (Ridge atau Lasso) di set terhadap L1 dan L2, sedangkan nilai C atau parameter penalty diset menggunakan rentang logspace(-5, 5, 20). Untuk model K Nearest Neighbor, nilai dari n_neighbor atau jumlah tetangga di set terhadap nilai 3, 5, 7, dan 9 dengan nilai weight Uniform dan Distance. Untuk model Decision Tree, nilai dari max depth untuk tree di set pada 2 sampai 12 dengan criterion gini, entrophy, dan logloss. Untuk model Support Vector Machine, parameter kernel di set menjadi linear, poly, dan rbf dengan nilai C sebesar logspace(-4, 4, 20). Berdasarkan experimentasi tersebut didapatkan *best parameter* sebagai berikut:

No	Model	Parameter
1	K-Nearest Neighbor	{'n_neighbors': 1, 'weights': 'uniform'}
2	Logistic Regression	{'C': 1e-05, 'penalty': 'l2'}
3	Support Vector Machine	Default
4	Decision Tree	{'criterion': 'gini', 'max_depth': 11}
5	Random Forest	Default

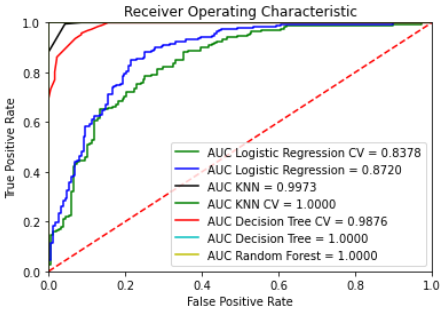
Berdasarkan metrics accuracy, didapatkan bahwa decision tree dan random forrest memiliki nilai accuracy 100%. Hal ini menunjukan potensi overfitting pada model tersebut

terhadap data training. Dari hasil eksperimentasi tersebut, didapatkan nilai akurasi, precision, recall dan f1 terhadap data training dan validation dari masing-masing percobaan sebagai berikut:

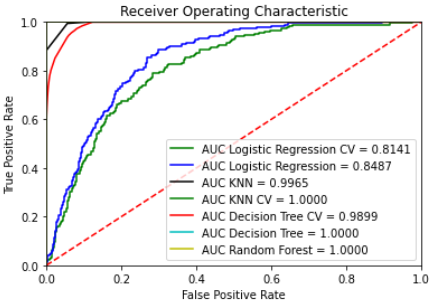
No	Model	Sampling Metode	Accuracy		Precision (Macro Avg)		Recall (Macro Avg)		F1-Score (Macro Avg)	
			Training	Validasi	Training	Validasi	Training	Validasi	Training	Validasi
1	K-Nearest Neighbor	Normal Sampling	0.896	1.000	0.660	1.000	0.950	1.000	0.710	1.000
2	Logistic Regression	Normal Sampling	0.746	0.951	0.560	0.480	0.770	0.500	0.540	0.490
3	Support Vector Machine	Normal Sampling	0.845	-	0.610	-	0.870	-	0.640	-
4	Decision Tree	Normal Sampling	1.000	0.983	1.000	0.960	1.000	0.850	1.000	0.890
5	Random Forest	Normal Sampling	1.000	-	1.000	-	1.000	-	1.000	-
6	K-Nearest Neighbor	Random Under Sampling	0.957	0.580	0.960	0.770	0.960	0.580	0.960	0.490
7	Logistic Regression	Random Under Sampling	0.797	0.500	0.800	0.250	0.800	0.500	0.800	0.330
8	Support Vector Machine	Random Under Sampling	0.885	-	0.890	-	0.890	-	0.890	-
9	Decision Tree	Random Under Sampling	1.000	0.848	1.000	0.880	1.000	0.850	1.000	0.840
10	Random Forest	Random Under Sampling	1.000	-	1.000	-	1.000	-	1.000	-
11	K-Nearest Neighbor	Random Over Sampling	0.946	0.578	0.950	0.760	0.950	0.580	0.950	0.490
12	Logistic Regression	Random Over Sampling	0.774	0.500	0.780	0.250	0.770	0.500	0.770	0.330
13	Support Vector Machine	Random Over Sampling	0.872	-	0.870	-	0.870	-	0.870	-
14	Decision Tree	Random Over Sampling	1.000	0.845	1.000	0.880	1.000	0.850	1.000	0.840
15	Random Forest	Random Over Sampling	1.000	-	1.000	-	1.000	-	1.000	-
16	K-Nearest Neighbor	Over Sampling (SMOTE)	0.944	0.609	0.950	0.770	0.940	0.610	0.940	0.540
17	Logistic Regression	Over Sampling (SMOTE)	0.795	0.500	0.800	0.250	0.800	0.500	0.790	0.330
18	Support Vector Machine	Over Sampling (SMOTE)	0.902	-	0.910	-	0.900	-	0.900	-
19	Decision Tree	Over Sampling (SMOTE)	1.000	0.653	1.000	0.790	1.000	0.650	1.000	0.610
20	Random Forest	Over Sampling (SMOTE)	1.000	-	1.000	-	1.000	-	1.000	-

Untuk model Support Vector Machine dan Random Forest tidak dilakukan Grid Search CV karena membutuhkan cost computation yang tinggi pada saat eksperimentasi sehingga membutuhkan waktu yang sangat lama untuk running. Lalu diputuskan untuk tidak melakukan Grid Search CV pada model tersebut.

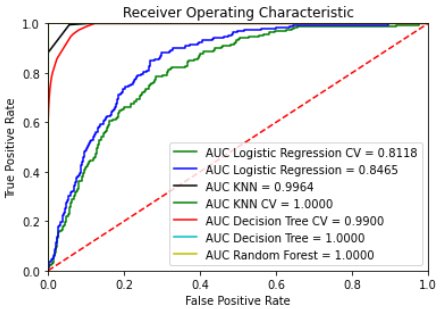
Selanjutnya dilakukan ploting terhadap ROC dan AUC terhadap konfigurasi data training dan validasi didapatkan nilai sebagai berikut:



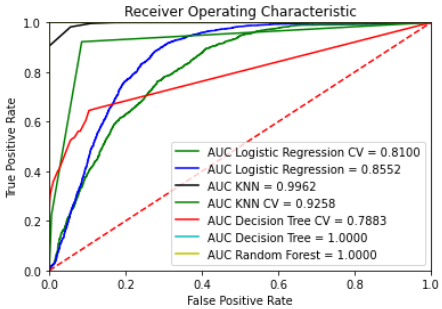
Random Under Sampling



Random Over Sampling



Normal Data



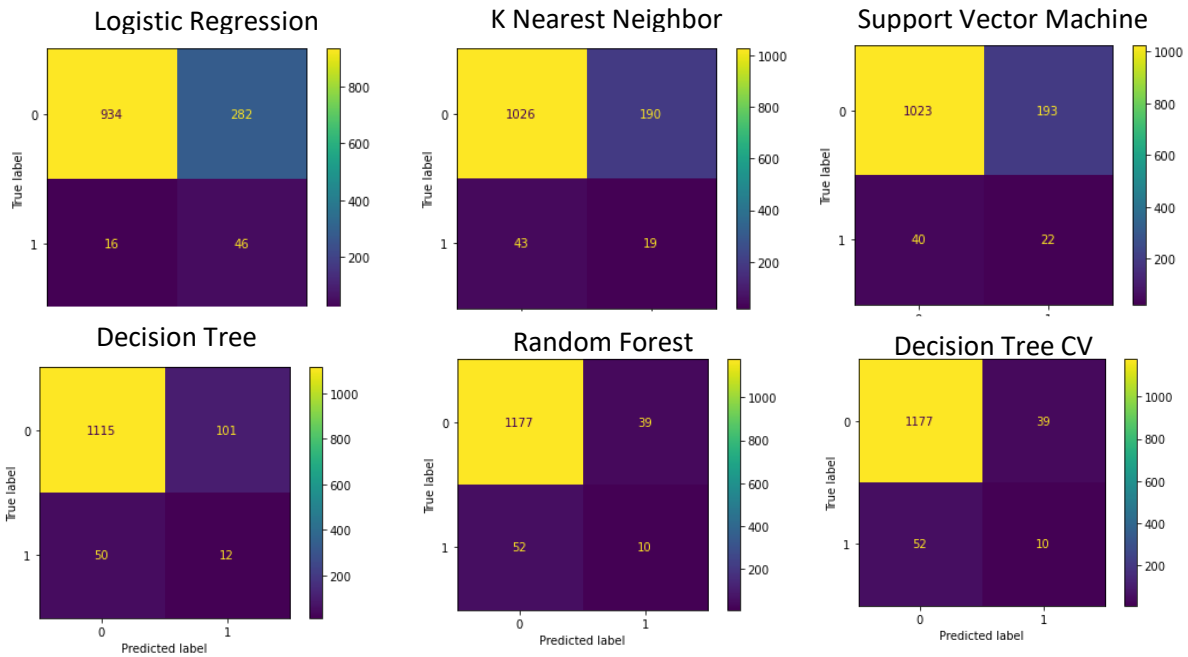
Over Sampling SMOTE

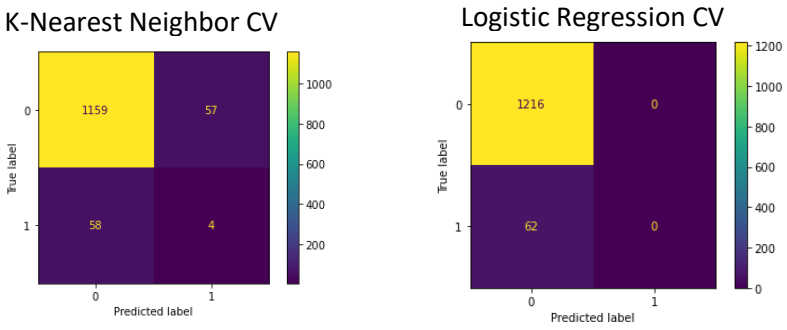
Berdasarkan eksperimentasi terhadap data training dan data validation tersebut didapatkan nilai AUC terbaik pada data Random Under Sampling. Hal tersebut wajar terjadi karena data yang diprediksi lebih sedikit sehingga tingkat kemungkinan benar prediksi lebih tinggi. Model Decision Tree dan Random Forest mengalami overfit sehingga dilakukan tree pruning dan menurukan nilai max depthnya menjadi 11.

Selanjutnya dilakukan proses fitting model pada data test untuk mendapatkan performance model. Model yang digunakan pada data test merupakan model yang sudah dibuat pada data training dan data validation. Berikut merupakan hasil dari performa model setelah permodelan pada data test:

No	Model	Accuracy	Precision (Macro Avg)	Recall (Macro Avg)	F1-Score (Macro Avg)	Area Under Curve (AUC)
		Testing	Testing	Testing	Testing	Testing
2	Logistic Regression	0.767	0.560	0.760	0.560	0.839
7	Logistic Regression - Cross Validation	0.952	0.480	0.500	0.490	0.812
5	Random Forest	0.911	0.520	0.520	0.520	0.769
8	Decision Tree - Cross Validation	0.929	0.580	0.560	0.570	0.687
1	K-Nearest Neighbor	0.818	0.530	0.580	0.520	0.612
3	Support Vector Machine	0.818	0.530	0.600	0.530	0.612
4	Decision Tree	0.882	0.530	0.560	0.540	0.555
6	K-Nearest Neighbor - Cross Validation	0.910	0.510	0.510	0.510	0.548

Berdasarkan hasil dari eksperimentasi tersebut, didapatkan bahwa Logistic Regresion memiliki Performa terbaik dalam hal Recall (0.760) dan AUC (0.839) sehingga model tersebut yang dipilih sebagai model terbaik walaupun dari sisi Accuracy hanya 0.767. Model selanjutnya yang dapat digunakan adalah model Logistic Regression CV dengan Accuracy 0.952, Recall 0.480, dan AUC 0.812. Namun model tersebut tidak dapat digunakan karena hanya memprediksi *non-stroke* saja. Selanjutnya dapat digunakan model ketiga yaitu model Random Forest dengan nilai Accuracy 0.911, Recall 0.520 dan AUC 0.839. Nilai nilai tersebut dapat tergambarkan pada Confusion Matrix berikut:





Dari eksperimentasi tersebut dapat disimpulkan bahwa akibat data yang imbalance >90% sehingga model-model tree terjadi sangat overfit terhadap data training. Sehingga model model tersebut tidak dapat merepresentasikan hasil prediksi yang baik pada data test walaupun sudah dilakukan tree pruning pada proses Cross Validation. Model logistic regression dapat dengan lebih baik mencapai nilai Recall dan AUC yang lebih tinggi.

6. Conclusion/Future Work

a. Metrics performansi Accuracy bukan metrics utama yang digunakan untuk mengukur performa model karena data imbalance > 95%. Sehingg digunakan metrics Recall dan AUC
b. Berdasarkan hasil eksperimentasi pada kasus ini didapatkan algoritma terbaik dalam proses Machine Learning Supervised adalah Logistic Regression dengan nilai Recall (0.760) dan AUC (0.839) dan Accuracy (0.767) pada data test. Sedangkan pada data training didapatkan nilai Recall (0.77), AUC (0.847), dan Accuracy (0.746).
c. Terjadi overfitting pada model tree yaitu Decision Tree dan Random Forest sehingga didapatkan Accuracy, Recall, dan AUC sebesar 1 untuk data training, tetapi mengalami penurunan signifikan pada data test sehingga perlu dilakukan tree pruning dan optimasi hyperparameter lebih lanjut untuk model berbasis tree tersebut.
d. Untuk eksperimentasi lebih lanjut, dapat digunakan Principal Component Analysis untuk mereduksi jumlah fiturnya sehingga dapat mengurangi kondisi overfitting dari model.
e. Dapat menggunakan model lain seperti Multi Layer Perceptron (Neural Network), Adaboost / XGBoost untuk membandingkan tingkat performansi model