**Laporan Data Mining**

**Permasalahan Klasifikasi**



**OLEH :**

**Nama : Afu Sidhi Pamekas**

**NPN : 17082010033**

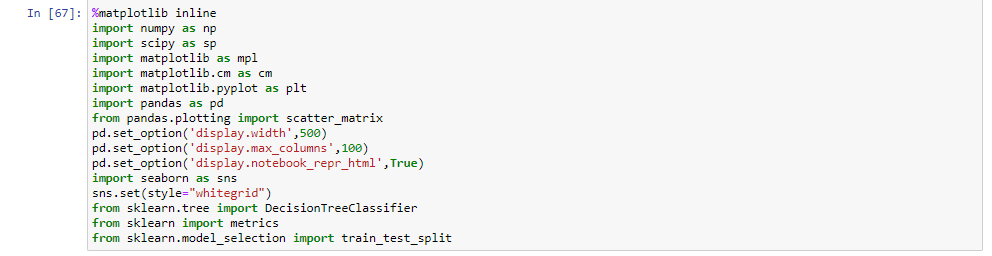
**Paralel : B**

**SISTEM INFORMASI**

**UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN” JAWA TIMUR**

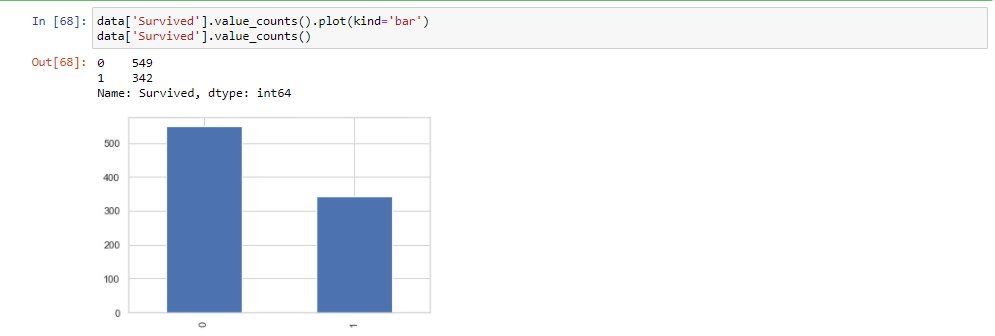
**SURABAYA**

**2020/2021**



## Pertama-tama untuk mengetahui apakah *dataset* yang digunakan *balance* atau *unbalance* diperlukan pengecekan rasio data dengan cara EDA (data yang digunakan data lanjutan modul 6 kemarin).

*Import* terlebih dahulu fungsi-fungsi yang diperlukan..



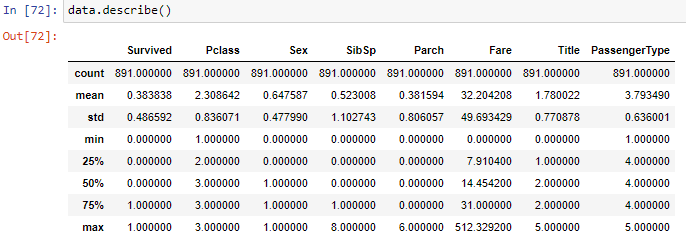
Masukkan code untuk membuat grafik seperti gambar ini. Jika dilihat sekilas dataset ini dilihat berdasarkan survive atau tidak survive relatif balance



Untuk lebih detailnya kita buat grafik seperti berikut. Sehingga didapat perbandingan angkanya, 0.090909 + 0.525253 = 0.616162 untuk meninggal dan 0.261504 + 0.122334 = 0.383838 untuk selamat

Jumlah korban titanic yang meninggal yaitu 0.616162 x 891 = 549, sedangkan korban yang selamat yaitu 0.383838 x 891 = 342. Maka jika dilihat rasionya 342 / 549 = 0.62295 = 0.6 atau jika diubah dalam bentuk pecahan 3/5 yang membuktikan bahwa *dataset* ini *balance.* (suatu *dataset* dikatakan *unbalance* jika rasio di bawah 1/4 atau 0.25)

Sebenarnya karena sudah *balance* kita tidak perlu melakukan *resampling.* Tapi katakanlah misalnya hasil tadi menyebutkan bahwa *dataset* tersebut *unbalance,* kita perlu melihat standar deviasi untuk menentukan cara *resampling* yang cocok untuk mengatasi hal tersebut.

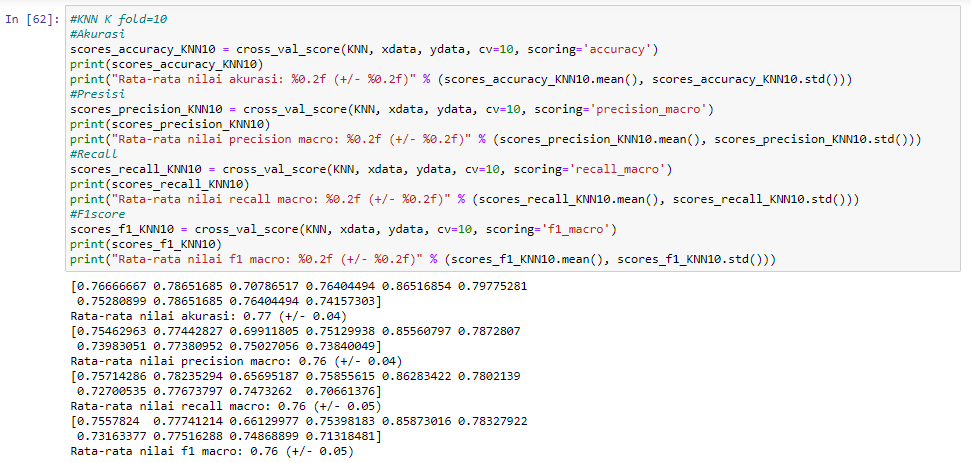


Untuk mencari standar deviasi menggunakan kode data.describe() yang akan menampilkan beberapa perhitungan sesuai dengan *dataset* yang kita berikan, dalam *describe* ini terdapat *count*, *mean*, std (standar deviasi), dll. Karena std relatif kecil yaitu 0.48 maka yang paling tepat yaitu melakukan *oversampling*. (sekali lagi *dataset* ini *balance* sehingga tidak diperlukan *resampling*)

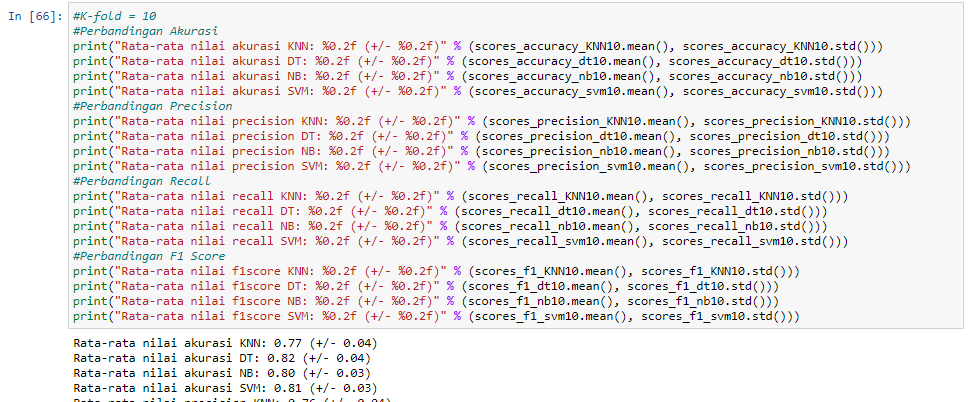
Selain itu untuk melihat apakah *dataset* tersebut benar-benar *balance* dapat juga digunakan dengan menggunakan algoritma NVM dan Naive Bayes (kedua algoritma ini tidak cocok untuk menangani *dataset* *imbalanced*), jika nilai *recall* jelek maka memang benar bahwa *dataset* ini *imbalance*

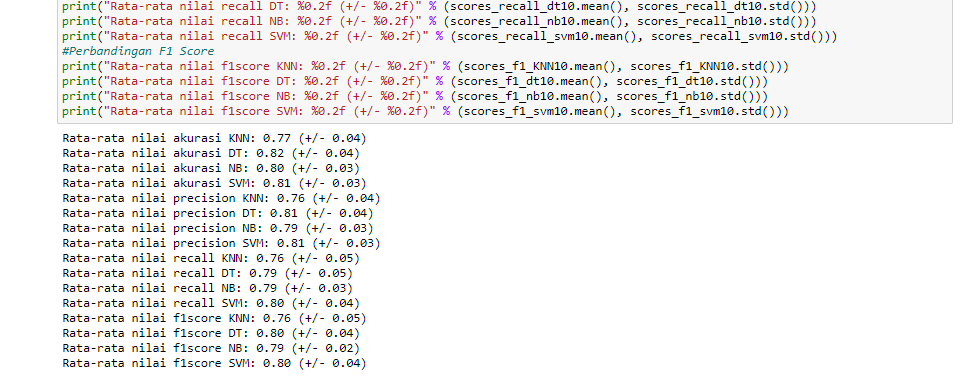
Untuk melihat perbandingan akurasi, presisi, *recall*, dan *f1-score* saya menggunakan holdout (training 70%, testing 30%) , crossval k-fold=5, dan crossval k-fold=10 dengan algoritma yang digunakan yaitu Decision Tree, NVM, Naive Bayes, dan KNN.

Pada holdout dan crossval k-fold=5 pada modul 6 sebelumnya sudah dibuat, tinggal menambahkan untuk crossval k-fold=10.



Untuk kode yang digunakan hanya berbeda pada cv=10 yang digunakan untuk menentukan jumlah k-fold dan variabel agar tidak tumpang tindih dengan variabel yang lain (lakukan juga pada algoritma yang lain).





Berikut hasil perbandingan jika k-fold=10. Agar lebih mudah dalam membandingkan saya tempatkan dalam bentuk tabel.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Akurasi | | | Precision | | |
| Algoritma | Holdout | Crossval K=5 | Crossval K=10 | Holdout | Crossval K=5 | Crossval K=10 |
| KNN | 0.76 | 0.78 | 0.77 | 0.77 | 0.76 | 0.72 |
| DT | 0.76 | 0.8 | 0.82 | 0.76 | 0.8 | 0.82 |
| NB | 0.76 | 0.8 | 0.8 | 0.77 | 0.79 | 0.8 |
| SVM | 0.77 | 0.81 | 0.81 | 0.77 | 0.8 | 0.81 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Recall | | | F1-Score | | |
| Algoritma | Holdout | Crossval K=5 | Crossval K=10 | Holdout | Crossval K=5 | Crossval K=10 |
| KNN | 0.84 | 0.76 | 0.76 | 0.8 | 0.76 | 0.76 |
| DT | 0.86 | 0.79 | 0.79 | 0.81 | 0.79 | 0.8 |
| NB | 0.82 | 0.79 | 0.79 | 0.79 | 0.79 | 0.79 |
| SVM | 0.85 | 0.8 | 0.8 | 0.81 | 0.8 | 0.8 |

Kesimpulannya dari tabel berikut algoritma Naive Bayes dan SVM masih bisa *menghandle* (hasil *recall* masih baik) *dataset* tersebut yang artinya memang benar bahwa *dataset* ini *balanced.*

*Source code* bisa dilihat di https://github.com/Afumoons/tugasdatmin/tree/week7