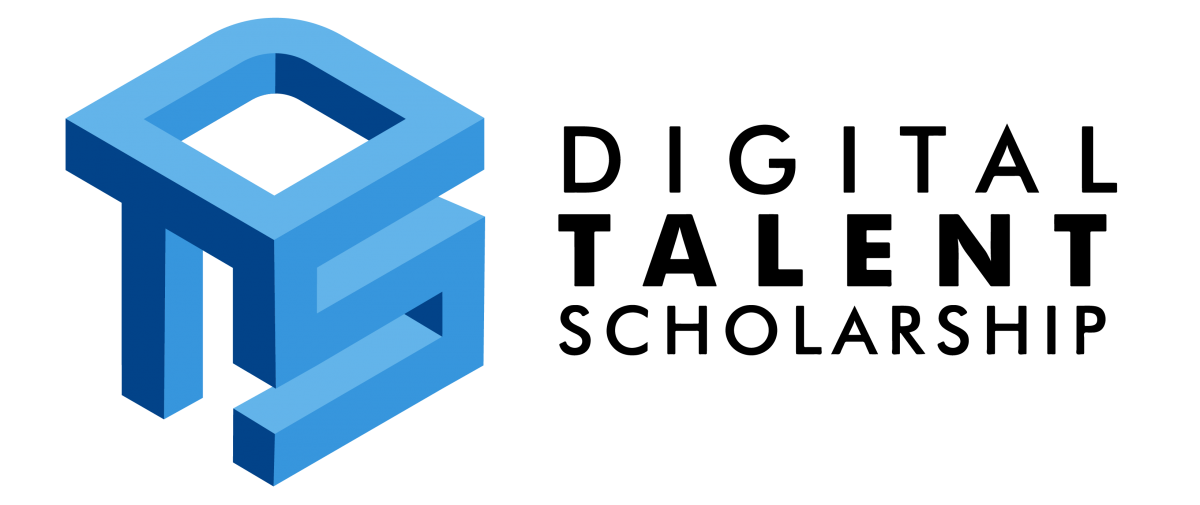
**Analisis Populeritas dari Suatu App**

**pada Google Play Store Apps**

****

Oleh:

Heri Setiawan

Muttaqien Rodhiya Robbi

**Program Artificial Intelligence**

**Kementrian Komunikasi dan Informatika**

**2019**

1. *Background*

Sampai sekarang ini, persentase aplikasi mobile terus meningkat. Android memegang sekitar 53,2% dari pasar ponsel cerdas, sementara iOS 43%. Untuk membuat lebih banyak orang mengunduh aplikasi, perlu memastikan mereka dapat dengan mudah menemukan aplikasi di *marketplace* (Google Play Store) . Analisis aplikasi seluler adalah cara terbaik untuk memahami strategi yang ada untuk mendorong pertumbuhan dan retensi pengguna di masa mendatang.

Biasanya aplikasi yang mudah dicari pada *marketplace* (Google Play Store) salah satunya adalah banyak unduhan dari suatu aplikasi tersebut. Dimana menjadi salah satu indikator terhadap kepopuleran suatu Aplikasi. Namun mengukur kepopuleran suatu aplikasi berdasarkan jumlah unduhan suatu aplikasi saja kurang tepat, hal ini terjadi karena banyak indikator lain yang berpengaruh terhadap kepopuleran aplikasi. Diantaranya yang berpengaruh terhadap kepopuleran suatu aplikasi adalah *rating* dari aplikasi, *review* dari aplikasi, *price* dari aplikasi dll.

Dengan jutaan aplikasi yang ada saat ini, kumpulan data berikut telah menjadi kunci untuk mendapatkan aplikasi yang sedang tren di toko aplikasi Google Play Store. Kumpulan data ini berisi lebih dari 10000 detail aplikasi seluler Android dengan dimensi dari dataset; 10.841 baris dan 13 kolom.

1. *Objective*

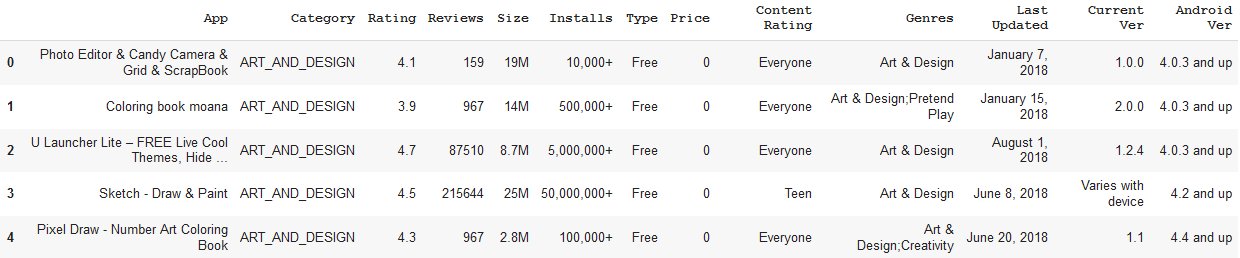
Dari ulasan di atas, tujuan dari Analisis ini dari data yang ada adalah mengklasifikasikan aplikasi berdasarkan kepopuleran. Selain itu, Analisis ini diharapkan dapat membantu para pemilik aplikasi agar dapat mudah mengetahui aplikasi mereka termasuk kedalam aplikasi pupuler atau tidak.

1. *Related Work (Literature Review)*
2. *Proposed Method*
3. Dataset

Dataset google play store didapatkan dari Kaggle yang terdiri dari 13 atributte, yaitu:

1. “App”, mencantumkan nama dari suatu Aplikasi,
2. “Category”, mencantumkan kategori dari suatu Aplikasi,
3. “Rating”, mencantumkan Rating dari suatu Aplikasi,
4. “Review”, mencantumkan banyaknya ulasan dari suatu Aplikasi,
5. “Size”, mencantumkan besarnya ukuran suatu Aplikasi,
6. “Installs”, banyaknya unduhan/install dari suatu Aplikasi,
7. “Type”, jenis dari suatu Aplikasi (Free/Paid),
8. “Price”, menampilkan harga dari suatu Aplikasi,
9. “Content Rating”, menampilkan diperuntukan /untuk siapa aplikasi dibuat,
10. “Genres”, hampir mirip dengan “Category” namun lebih spesifik dengan menampilkan sub dari “Category” suatu Aplikasi.
11. “Last Updated”, menampilkan terakhir kali aplikasi di perbaharui,
12. “Current Ver”, menampilkan versi terakhir dari Aplikasi,
13. “Android Ver”, menampilakan batas versi minimum dari android yang bisa dijalankan dari suatu Aplikasi.

Berikut ini contoh tampilan dataset yang digunakan



Gambar 1. Contoh dataset yang digunakan

Dataset ini memiliki *missing values,* dan banyak sekali data yang tidak sesuai dengan tipe data seharusnya. Sehingga dibutuhkan penanganan khusus, dalam eksperimen ini *missing values* diisi dengan nilai rata-rata dari setiap kolomnya masing-masing dan juga mengubah tipe data agar menjadi tipe data yang seharusnya.

1. Arsitektur

*Data Understanding*

*Pre-Processing* Data

*Modeling*

Gambar 2. Arsitektur Program untuk Klasifikasi Popularity

1. *Data Understanding*

Proses ini adalah proses memahami data dan apa yang seharusnya dilakukan perubahan pada data, yaitu:

1. Data memiliki 12 tipe objek dan satu tipe numerik.
2. Data “Size”, bertipe objek karena terdapat satuan kilo (k) dan Mega (M). Sehingga harus dilakukan perubahan dengan nilai yang sesuai dengan besaranya. Selain itu terdapat data berisi “Varies with device” yang diganti dengan NaN dan mengkonvert 1000+ menjadi numerik 1000.
3. Data “Installs”, bertipe objek karena terdapat tanda “+” dan “free” string.
4. Data “Category”, ada satu data unik “1.9” karena kesalahan input pada data.
5. *Pre-Processing*  Data

Seperti yang sudah dijelaskan di atas. Maka dilakukan beberapa proses, yaitu:

1. Pada data “Size” hilangkan “k” dan “M” dan juga “Varies with device”. Kemudian mengganti data kosong dengan “NaN”. Setelah itu mengganti data tipe nya menjadi “float64”. Untuk data “NaN” akan dihilangkan.
2. Pada data “Installs” hilangkan tanda “+”, mengganti tanda “,” dengan mengkosongkan/menghilangkannya. Kemudian mengganti tipe data menjadi “int64”.
3. Untuk data “Price” menghilangkan tanda “$”, kemudian mengganti tipe data menjadi “float64”.
4. Karena kami akan mengkategorikan Apps berdasarkan Popularitas yang berdasarkan jumlah Installs. Maka dibuat *feature* baru dengan label “Popularity” dimana jika installs < 50 *Not Popular,* 50≤ *Less Popular* < 5000, 5000 ≤ *Ordinary* < 500000, 500000 ≤ *Popular* < 50000000, lainnya *Very Popular.*
5. Data “App” kami hilangkan karena unik, Data “Category” dan juga “Installs” kami hilangkan karena “Popularity” berdasarkan data “Installs”.
6. *Modeling*
7. KNeighborsClassifier

Adalah *​supervised classifier* yang membuat model dengan cara melihat sejumlah k tetangga terdekat dari setiap datanya.

1. DecisionTreeClassifier

Adalah sebuah diagram alir yang berbentuk seperti pohon yang memiliki sebuah *root node* yang digunakan untuk mengumpulkan data.

1. *Random Forest*

Adalah kombinasi dari  masing–masing *tree* yang baik kemudian dikombinasikan  ke dalam satu model.

1. Skenario Eksperimen
2. Tahap *Pre Processing* Data. pada tahap ini, data yang telah dibersihkan akan dicari hasil kinerja menggunakan tiga jenis model, dengan perlakuan terhadap data NaN:
3. Pada data “Size” di hapus.
4. Pada data “Size” diganti dengan *mean*.
5. Tahap *Processing* Data. Pada tahap ini, akan dibandingkan hasil kinerja dengan data sebelum di-*Balancing* (9145) dan sesudah di-*Balancing* ()*.*
6. Tahap Pemilihan Model. Pada tahap ini menggunakan tiga jenis model dan dipilih model dan parameter yang menghasilkan hasil klasifikasi paling baik.
7. Hasil Eksperimen
8. Tahap *Pre Processig* Data
9. Data NaN pada data “Size” dihapus*.*

Tabel 1. Nilai Hasil Eksperimen Variasi Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | *F1 Score* | *Accuracy Score* | *Precision Score* | *Recall Score* |
| KNN | 0.72 | 0.78 | 0.73 | 0.72 |
| DecisionTree | 0.74 | 0.81 | 0.75 | 0.73 |
| Random Forest | 0.78 | 0.84 | 0.80 | 0.77 |

1. Data NaN pada data “Size” di ganti *mean*D

Tabel 2. Nilai Hasil Eksperimen dengan Variasi Model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | *F1 Score* | *Accuracy Score* | *Precision Score* | *Recall Score* |
| KNN | 0.72 | 0.78 | 0.73 | 0.72 |
| DecisionTree | 0.74 | 0.81 | 0.75 | 0.73 |
| Random Forest | 0.78 | 0.84 | 0.80 | 0.77 |

Data yang telah dibersihkan dilakukan uji kinerja menggunakan tiga jenis Model. Hasil dari eksperimen dengan beberapa variasi model dapat dapat dilihat pada tabel diatas.

1. Tahap *Processing* Data

Tabel 3. Hasil Eksperimen data sebelum di Balancing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | *F1 Score* | *Accuracy Score* | *Precision Score* | *Recall Score* |
| KNN | 0.72 | 0.78 | 0.73 | 0.72 |
| DecisionTree | 0.74 | 0.81 | 0.75 | 0.73 |
| Random Forest | 0.78 | 0.84 | 0.80 | 0.77 |

Tabel 4. Hasil Eksperimen data setelah di Balancing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | *F1 Score* | *Accuracy Score* | *Precision Score* | *Recall Score* |
| KNN | 0.72 | 0.78 | 0.73 | 0.72 |
| DecisionTree | 0.74 | 0.81 | 0.75 | 0.73 |
| Random Forest | 0.78 | 0.84 | 0.80 | 0.77 |

Perbandingan data hasil eksperimen data sebelum dan sesudah di *Balancing* dapat dilihat pada tabel di atas.

1. Tahap *Modeling*
2. Analisis

Untuk data preprocessing, perlakuan terhadap data NaN pada data “Size” menggantinya dengan data *mean* dapat meningkatkan akurasi yang lebih baik.

Untuk tahap processing, data yang belum di *balancing* lebih baik dari data setelah di *Balancing*.

References

* http://machinelearning.mipa.ugm.ac.id/2018/07/28/random-forest/