

KLASIFIKASI

**PREDIKSI KEBERHASILAN APLIKASI *MOBILE* DENGAN DATASET
“GOOGLEPLAYSTORE”**

LAPORAN

Oleh:

Tim GCD

IF-A PAGI

- **211110347 - CINDY SINTIYA**
- **211110948 - GRACE HELENA HUTAGAOL**
- **211111930 - DAVID BATE'E**



UNIVERSITAS
MIKROSKIL

**PROGRAM STUDI S-1
FAKULTAS INFORMATIKA
TEKNIK INFORMATIKA
UNIVERITAS MIKROSKIL
2024/2025**

Judul

Prediksi Keberhasilan Aplikasi *Mobile* dengan Dataset “GooglePlayStore”
Menggunakan Metode Klasifikasi

Tujuan

1. **Mengoptimalkan pengambilan keputusan *app developer*:** Memberikan wawasan kepada *developer* mengenai fitur atau karakteristik aplikasi yang berpotensi meningkatkan keberhasilan aplikasi di pasar berdasarkan hasil prediksi model.
2. **Menguji pengaruh variabel terhadap kesuksesan aplikasi:** Mengidentifikasi dan menganalisis variabel atau fitur dalam dataset Google PlayStore, seperti jenis aplikasi (gratis atau berbayar), *rating*, atau kategori lain, terhadap potensi kesuksesan aplikasi di pasar.

Sumber Data

[kaggle.com/datasets/zeshanali/googleplaystore](https://www.kaggle.com/datasets/zeshanali/googleplaystore)

Fitur Independen (x)

- | | |
|--------------------------|---|
| 1. <i>App</i> | : nama aplikasi |
| 2. <i>Category</i> | : kategori aplikasi |
| 3. <i>Reviews</i> | : jumlah pemberi <i>Rating</i> |
| 4. <i>Size</i> | : ukuran aplikasi |
| 5. <i>Installs</i> | : jumlah pengunduh |
| 6. <i>Type</i> | : jenis aplikasi (gratis atau berbayar) |
| 7. <i>Price</i> | : harga aplikasi |
| 8. <i>Content Rating</i> | : <i>range</i> umur dari pemberi <i>Rating</i> |
| 9. <i>Genres</i> | : mirip seperti kategori aplikasi |
| 10. <i>Last Updated</i> | : tanggal terakhir aplikasi di- <i>update</i> |
| 11. <i>Current Ver</i> | : kode versi terbaru aplikasi |
| 12. <i>Android Ver</i> | : versi Android minimal agar dapat mengunduh aplikasi |

Fitur Dependen (y)

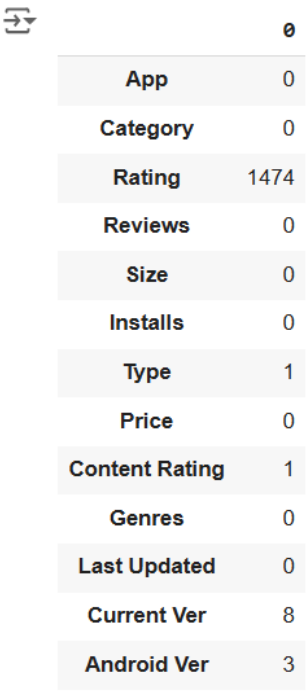
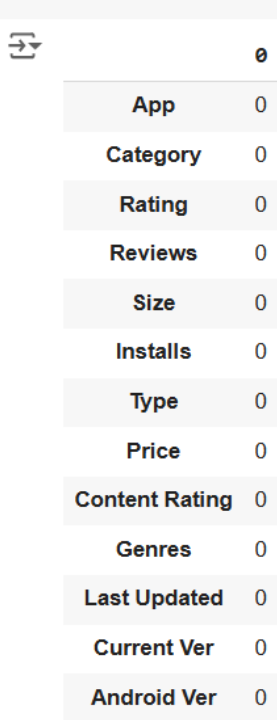
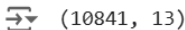

Rating

Ukuran Dataset

10.841 baris data, 13 kolom fitur

Preprocessing

1. Menghapus row yang berisi nilai “NaN”

Before	After
<pre># cek apakah ada dataset yang hilang raw_data.isna().sum()</pre>  <p>dtype: int64</p>	<pre># hapus baris data yg bernilai nan preprocessed_data = raw_data.dropna() preprocessed_data.isna().sum()</pre>  <p>dtype: int64</p>
<pre>raw_data.shape</pre>  <p>(10841, 13)</p>	<pre>preprocessed_data.shape</pre>  <p>(9360, 13)</p>

2. Perbaiki tipe data

Beberapa fitur yang seharusnya bertipe numerik, seperti *price*/ harga dan *installs*/ jumlah pengunduh namun tercatat sebagai *string* karena mengandung simbol akan diperbaiki.

```
# perbaiki format data numeric-string menjadi numerik
preprocessed_data['Price'] = preprocessed_data['Price'].replace('[\$,]', '', regex=True).astype(float)
preprocessed_data['Installs'] = preprocessed_data['Installs'].replace('[\+,]', '', regex=True).astype(int)

[ ] # transform string numeric data to integer

for i in preprocessed_data.columns:
    if preprocessed_data[i].dtype == 'object':
        try:
            preprocessed_data[i] = preprocessed_data[i].astype('int64')
        except:
            continue
```

Before	After
<pre># cek struktur data raw_data.info()</pre> <pre><class 'pandas.core.frame.DataFrame'> RangeIndex: 10841 entries, 0 to 10840 Data columns (total 13 columns): # Column Non-Null Count Dtype --- - 0 App 10841 non-null object 1 Category 10841 non-null object 2 Rating 9367 non-null float64 3 Reviews 10841 non-null object 4 Size 10841 non-null object 5 Installs 10841 non-null object 6 Type 10840 non-null object 7 Price 10841 non-null object 8 Content Rating 10840 non-null object 9 Genres 10841 non-null object 10 Last Updated 10841 non-null object 11 Current Ver 10833 non-null object 12 Android Ver 10838 non-null object dtypes: float64(1), object(12) memory usage: 1.1+ MB</pre>	<pre># cek kembali tipe data per kolom preprocessed_data.info()</pre> <pre><class 'pandas.core.frame.DataFrame'> Index: 9360 entries, 0 to 10840 Data columns (total 13 columns): # Column Non-Null Count Dtype --- - 0 App 9360 non-null object 1 Category 9360 non-null object 2 Rating 9360 non-null float64 3 Reviews 9360 non-null int64 4 Size 9360 non-null object 5 Installs 9360 non-null int64 6 Type 9360 non-null object 7 Price 9360 non-null float64 8 Content Rating 9360 non-null object 9 Genres 9360 non-null object 10 Last Updated 9360 non-null object 11 Current Ver 9360 non-null object 12 Android Ver 9360 non-null object dtypes: float64(2), int64(2), object(9) memory usage: 1023.8+ KB</pre>

3. Pengelompokkan/ *Binning* kelas

Karena kolom yang akan ditetapkan menjadi kelas memiliki jumlah yang terlalu banyak,

```
Nama kolom: Rating (39, float64)
[4.1 3.9 4.7 4.5 4.3 4.4 3.8 4.2 4.6 4.  4.8 4.9 3.6 3.7 3.2 3.3 3.4 3.5
 3.1 5.  2.6 3.  1.9 2.5 2.8 2.7 1.  2.9 2.3 2.2 1.7 2.  1.8 2.4 1.6 2.1
 1.4 1.5 1.2]
```

kami memutuskan untuk melakukan pengelompokkan kelas berdasarkan *range rating* tertentu atau disebut dengan *Binning* (teknik mengelompokkan nilai-nilai numerik/ kontinu menjadi nilai kategoris/ diskrit).

<pre># class binning (grouping continuous numerical data to discrete classes) preprocessed_data['Class'] = 3 preprocessed_data.loc[preprocessed_data['Rating'] < 4, 'Class'] = 2 preprocessed_data.loc[preprocessed_data['Rating'] < 2, 'Class'] = 1 preprocessed_data.head()</pre>	<pre># Count the number of occurrences of each class class_counts = preprocessed_data['Class'].value_counts() classes = sorted(class_counts.keys()) class_counts</pre>
<pre>App Category Rating Reviews Size Installs Type Price Content Rating Genres Last Updated Current Ver Android Ver Class 0 Photo Editor & Candy Camera & Grid & ScrapBook ART_AND_DESIGN 4.1 159 19M 10000 Free 0.0 Everyone Art & Design 7-Jan-18 1.0.0 4.0.3 and up 3 1 Coloring book moana ART_AND_DESIGN 3.9 967 14M 500000 Free 0.0 Everyone Art & Design;Pretend Play 15-Jan-18 2.0.0 4.0.3 and up 2 2 U Launcher Lite – FREE Live Cool Themes, Hide... ART_AND_DESIGN 4.7 87510 8.7M 5000000 Free 0.0 Everyone Art & Design 1-Aug-18 1.2.4 4.0.3 and up 3 3 Sketch - Draw & Paint ART_AND_DESIGN 4.5 215644 25M 50000000 Free 0.0 Teen Art & Design 8-Jun-18 Varies with device 4.2 and up 3 4 Pixel Draw - Number Art Coloring Book ART_AND_DESIGN 4.3 967 2.8M 100000 Free 0.0 Everyone Art & Design;Creativity 20-Jun-18 1.1 4.4 and up 3</pre>	<pre>count Class 3 7363 2 1941 1 56 dtype: int64</pre>

4. *Encoding* fitur

Tipe data dari masing-masing kolom fitur masih berupa *string* dan numerik, sehingga perlu dilakukan *encoding* untuk mengubah *value* dari *string* kategori menjadi nilai numerik agar dapat diolah oleh model.

Untuk *App*/ nama aplikasi yang bersifat *unique* akan dijadikan sebagai *index* data agar tidak ikut di-*encode*.

```

label_mappings = {} # init dictionary to store label encode

preprocessed_data = preprocessed_data.set_index('App') # skipping app name encoding

for col in preprocessed_data.columns:
    if preprocessed_data[col].dtype == 'object':
        try:
            # encode string to numeric feature
            label_encoder = LabelEncoder()
            preprocessed_data[col] = label_encoder.fit_transform(preprocessed_data[col])
            label_mappings[col] = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))
        except:
            continue

preprocessed_data.head()

```

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	Class
Photo Editor & Candy Camera & Grid & ScrapBook	0	4.1	159	47	10000	0	0.0	1	9	1195	104	14	3
Coloring book moana	0	3.9	967	24	500000	0	0.0	1	11	284	932	14	2
U Launcher Lite - FREE Live Cool Themes, Hide Apps	0	4.7	87510	333	5000000	0	0.0	1	9	6	419	14	3
Sketch - Draw & Paint	0	4.5	215644	87	50000000	0	0.0	4	9	1241	2538	17	3
Pixel Draw - Number Art Coloring Book	0	4.3	967	56	100000	0	0.0	1	10	544	246	19	3

Berikut contoh label *encode* dan tipe data akhir dari proses *preprocessing*:

```

for col, mapping in label_mappings.items():
    print(f"\nMapping for column '{col}':")
    display(pd.DataFrame(list(mapping.items()), columns=['Original', 'Encoded']))

```

Mapping for column 'Type':

Original	Encoded
0 Free	0
1 Paid	1

Mapping for column 'Content Rating':

Original	Encoded
0 Adults only 18+	0
1 Everyone	1
2 Everyone 10+	2
3 Mature 17+	3
4 Teen	4
5 Unrated	5

```

# cek kembali tipe data per kolom
preprocessed_data.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Index: 9360 entries, Photo Editor & Candy Camera
Data columns (total 13 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   Category        9360 non-null   int64
1   Rating          9360 non-null   float64
2   Reviews         9360 non-null   int64
3   Size            9360 non-null   int64
4   Installs        9360 non-null   int64
5   Type            9360 non-null   int64
6   Price           9360 non-null   float64
7   Content Rating  9360 non-null   int64
8   Genres          9360 non-null   int64
9   Last Updated    9360 non-null   int64
10  Current Ver     9360 non-null   int64
11  Android Ver     9360 non-null   int64
12  Class           9360 non-null   int64
dtypes: float64(2), int64(11)
memory usage: 1.3+ MB

```

Pembagian Train-Test

Data *training* dan *testing* dibagi dalam rasio 7 : 3, dimana 70% dataset hasil *preprocessing* akan digunakan dalam proses *training* dan sisa 30% akan dipakai sebagai data *testing* untuk evaluasi akurasi model yang dibangun. Beberapa kolom fitur dihapus setelah beberapa kali membangun model untuk mendapatkan hasil terbaik. Kolom *Rating* juga dihapus karena penentuan kelas sudah menggunakan kolom hasil *binning*.

```

# memisahkan variabel x dan y dari tabel
X_data = preprocessed_data.drop(['Rating', 'Genres', 'Class', 'Last Updated', 'Current Ver', 'Android Ver'], axis = 1)
y_data = preprocessed_data['Class'].astype(str)

```

```

[ ] # bagi data untuk training n testing dgn rasio 7:3

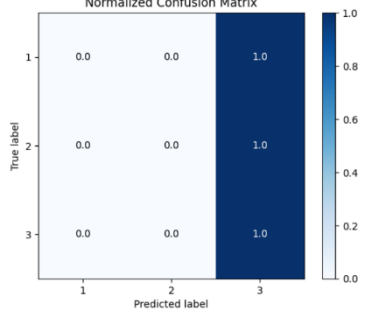
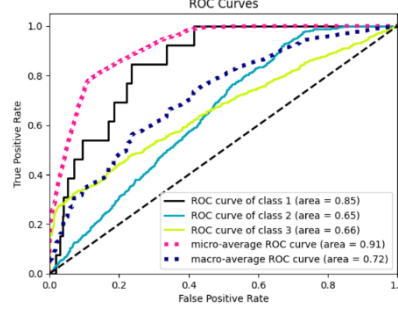
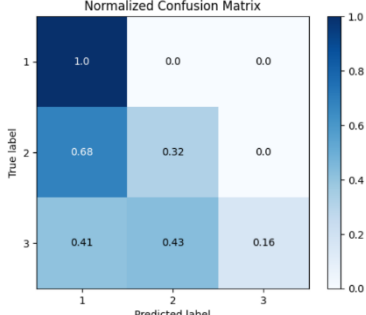
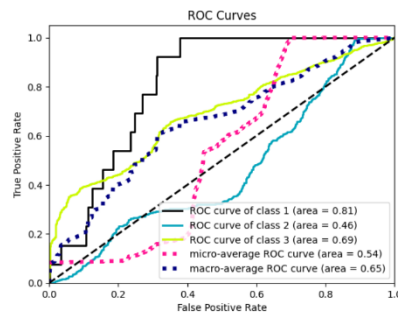
X_train, X_test, y_train, y_test = train_test_split(
    X_data,
    y_data,
    test_size = 0.3,
    random_state = 1000
)

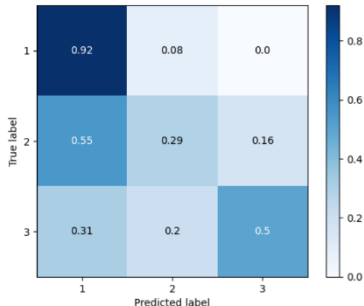
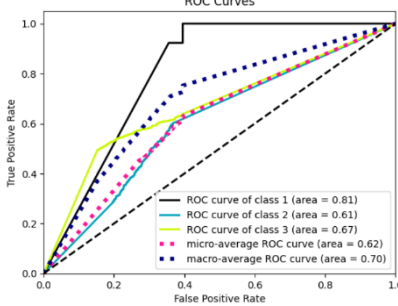
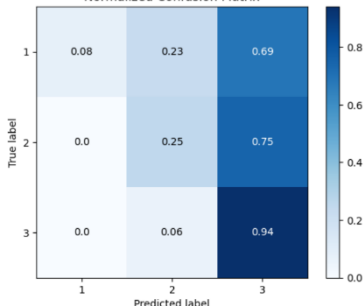
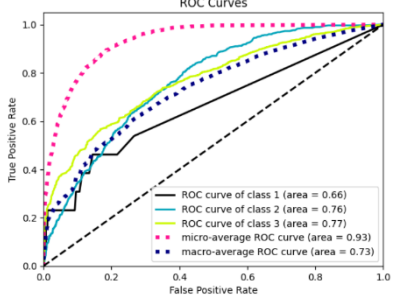
```

Pemilihan Model

<p>Logistic Regression</p> <pre>logreg_model = LogisticRegression(max_iter=200) logreg_model.fit(X_train, y_train)</pre> 	<p>Naive Bayes (Gaussian)</p> <pre>nb_model = GaussianNB() nb_model.fit(X_train, y_train)</pre> 
<p>Random Forest</p> <pre>randfor_model = RandomForestClassifier(n_estimators=900, random_state=1000, randfor_model.fit(X_train, y_train)</pre> 	<p>Naive Bayes (Multinomial)</p> <pre>nb_model = MultinomialNB() nb_model.fit(X_train, y_train)</pre> 

Evaluasi Model

<div>Logistic Regression</div>	<div><p>Normalized Confusion Matrix</p></div> <div><p>ROC Curves</p></div>																																			
	<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>accuracy</th><th>macro avg</th><th>weighted avg</th></tr><tr><td>precision</td><td>0.0</td><td>1.000000</td><td>0.781540</td><td>0.781695</td><td>0.593847</td><td>0.824756</td></tr><tr><td>recall</td><td>0.0</td><td>0.003322</td><td>1.000000</td><td>0.781695</td><td>0.334441</td><td>0.781695</td></tr><tr><td>f1-score</td><td>0.0</td><td>0.006623</td><td>0.877375</td><td>0.781695</td><td>0.294666</td><td>0.686635</td></tr><tr><td>support</td><td>13.0</td><td>602.000000</td><td>2193.000000</td><td>0.781695</td><td>2808.000000</td><td>2808.000000</td></tr></table>		1	2	3	accuracy	macro avg	weighted avg	precision	0.0	1.000000	0.781540	0.781695	0.593847	0.824756	recall	0.0	0.003322	1.000000	0.781695	0.334441	0.781695	f1-score	0.0	0.006623	0.877375	0.781695	0.294666	0.686635	support	13.0	602.000000	2193.000000	0.781695	2808.000000	2808.000000
	1	2	3	accuracy	macro avg	weighted avg																														
precision	0.0	1.000000	0.781540	0.781695	0.593847	0.824756																														
recall	0.0	0.003322	1.000000	0.781695	0.334441	0.781695																														
f1-score	0.0	0.006623	0.877375	0.781695	0.294666	0.686635																														
support	13.0	602.000000	2193.000000	0.781695	2808.000000	2808.000000																														
<div>Naive Bayes – Gaussian</div>	<div><p>Normalized Confusion Matrix</p></div> <div><p>ROC Curves</p></div>																																			

	<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>accuracy</th><th>macro avg</th><th>weighted avg</th></tr><tr><td>precision</td><td>0.009878</td><td>0.168421</td><td>0.991477</td><td>0.197293</td><td>0.389926</td><td>0.810480</td></tr><tr><td>recall</td><td>1.000000</td><td>0.318937</td><td>0.159143</td><td>0.197293</td><td>0.492693</td><td>0.197293</td></tr><tr><td>f1-score</td><td>0.019564</td><td>0.220436</td><td>0.274263</td><td>0.197293</td><td>0.171421</td><td>0.261544</td></tr><tr><td>support</td><td>13.000000</td><td>602.000000</td><td>2193.000000</td><td>0.197293</td><td>2808.000000</td><td>2808.000000</td></tr></table>		1	2	3	accuracy	macro avg	weighted avg	precision	0.009878	0.168421	0.991477	0.197293	0.389926	0.810480	recall	1.000000	0.318937	0.159143	0.197293	0.492693	0.197293	f1-score	0.019564	0.220436	0.274263	0.197293	0.171421	0.261544	support	13.000000	602.000000	2193.000000	0.197293	2808.000000	2808.000000	
	1	2	3	accuracy	macro avg	weighted avg																															
precision	0.009878	0.168421	0.991477	0.197293	0.389926	0.810480																															
recall	1.000000	0.318937	0.159143	0.197293	0.492693	0.197293																															
f1-score	0.019564	0.220436	0.274263	0.197293	0.171421	0.261544																															
support	13.000000	602.000000	2193.000000	0.197293	2808.000000	2808.000000																															
Naive Bayes – Multinomial	<div><div><p>Normalized Confusion Matrix</p></div><div><p>ROC Curves</p></div></div>	<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>accuracy</th><th>macro avg</th><th>weighted avg</th></tr><tr><td>precision</td><td>0.011799</td><td>0.285714</td><td>0.918782</td><td>0.452991</td><td>0.405432</td><td>0.778861</td></tr><tr><td>recall</td><td>0.923077</td><td>0.289037</td><td>0.495212</td><td>0.452991</td><td>0.569109</td><td>0.452991</td></tr><tr><td>f1-score</td><td>0.023301</td><td>0.287366</td><td>0.643556</td><td>0.452991</td><td>0.318074</td><td>0.564321</td></tr><tr><td>support</td><td>13.000000</td><td>602.000000</td><td>2193.000000</td><td>0.452991</td><td>2808.000000</td><td>2808.000000</td></tr></table>		1	2	3	accuracy	macro avg	weighted avg	precision	0.011799	0.285714	0.918782	0.452991	0.405432	0.778861	recall	0.923077	0.289037	0.495212	0.452991	0.569109	0.452991	f1-score	0.023301	0.287366	0.643556	0.452991	0.318074	0.564321	support	13.000000	602.000000	2193.000000	0.452991	2808.000000	2808.000000
		1	2	3	accuracy	macro avg	weighted avg																														
precision	0.011799	0.285714	0.918782	0.452991	0.405432	0.778861																															
recall	0.923077	0.289037	0.495212	0.452991	0.569109	0.452991																															
f1-score	0.023301	0.287366	0.643556	0.452991	0.318074	0.564321																															
support	13.000000	602.000000	2193.000000	0.452991	2808.000000	2808.000000																															
Random Forest	<div><div><p>Normalized Confusion Matrix</p></div><div><p>ROC Curves</p></div></div>	<table><tr><th></th><th>1</th><th>2</th><th>3</th><th>accuracy</th><th>macro avg</th><th>weighted avg</th></tr><tr><td>precision</td><td>0.333333</td><td>0.517361</td><td>0.816845</td><td>0.785613</td><td>0.555847</td><td>0.750401</td></tr><tr><td>recall</td><td>0.076923</td><td>0.247508</td><td>0.937528</td><td>0.785613</td><td>0.420653</td><td>0.785613</td></tr><tr><td>f1-score</td><td>0.125000</td><td>0.334831</td><td>0.873036</td><td>0.785613</td><td>0.444289</td><td>0.754189</td></tr><tr><td>support</td><td>13.000000</td><td>602.000000</td><td>2193.000000</td><td>0.785613</td><td>2808.000000</td><td>2808.000000</td></tr></table>		1	2	3	accuracy	macro avg	weighted avg	precision	0.333333	0.517361	0.816845	0.785613	0.555847	0.750401	recall	0.076923	0.247508	0.937528	0.785613	0.420653	0.785613	f1-score	0.125000	0.334831	0.873036	0.785613	0.444289	0.754189	support	13.000000	602.000000	2193.000000	0.785613	2808.000000	2808.000000
		1	2	3	accuracy	macro avg	weighted avg																														
precision	0.333333	0.517361	0.816845	0.785613	0.555847	0.750401																															
recall	0.076923	0.247508	0.937528	0.785613	0.420653	0.785613																															
f1-score	0.125000	0.334831	0.873036	0.785613	0.444289	0.754189																															
support	13.000000	602.000000	2193.000000	0.785613	2808.000000	2808.000000																															

Hyperparameter Tuning

Kami memilih model Random Forest yang memiliki akurasi terbaik untuk dilakukan proses *hyperparameter tuning* agar mendapatkan *hyperparameter* terbaik untuk model terbaik tersebut.

```
# Mendefinisikan model random forest
randfor_model = RandomForestClassifier()

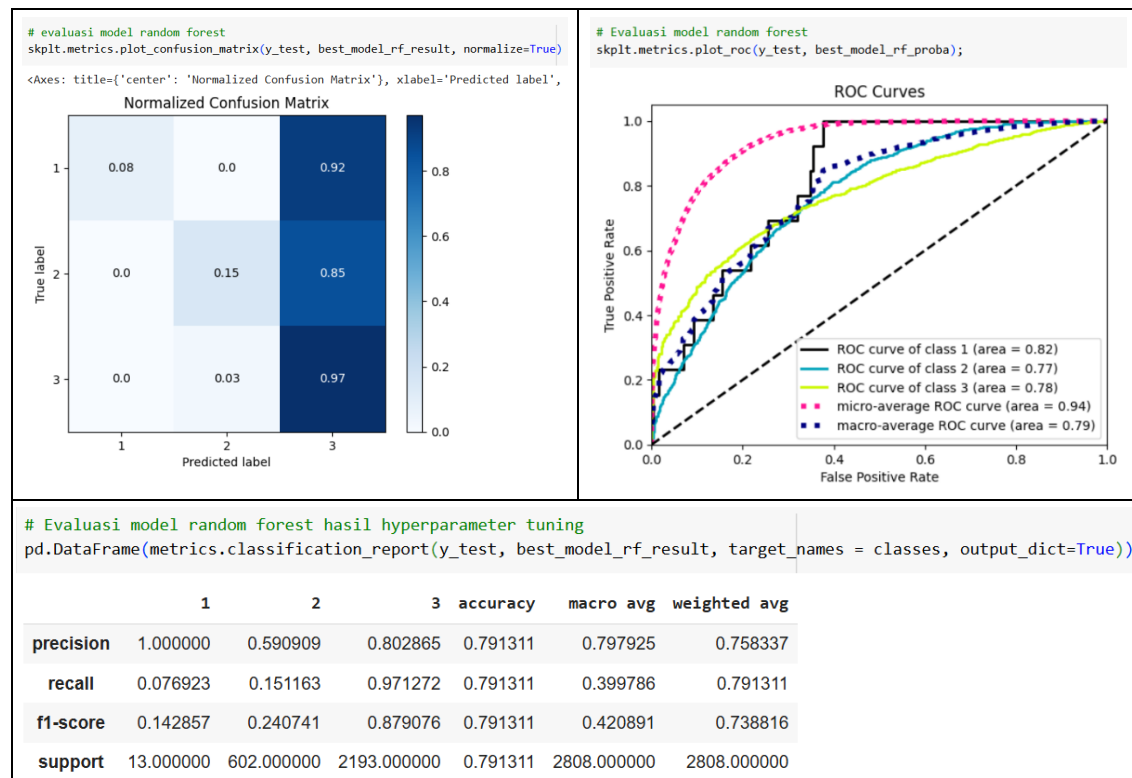
# mendefinisikan parameter untuk tuning
param_grid = {
    'n_estimators': [500, 750, 900],          # Number of trees in the forest
    'max_depth': [15, 35, None],              # Maximum depth of the tree
    'random_state': [42, 1000],               # Set seed for reproducibility
}

# definisikan objek GridSearchCV
grid_search = GridSearchCV(
    estimator=randfor_model,
    param_grid=param_grid,
    cv=3,
    scoring='accuracy'
)

# fit model dengan data
grid_search.fit(X_train, y_train)
```

```
GridSearchCV
best_estimator_: RandomForestClassifier
RandomForestClassifier
RandomForestClassifier(max_depth=15, n_estimators=750, random_state=42)
```

Dan hasilnya adalah akurasi model sedikit meningkat dari 78,56% menjadi 79,13%.



Interpretasi Model

Model dengan label Random Forest yang dibangun menggunakan *library* sklearn dan metode klasifikasi, menerima data *testing* sebanyak 2.808 baris data dan 7 kolom fitur. Nilai residual atau *error* model berkisar antara 0,524 hingga 3,0 yang menandakan tingkat *error* model masih cukup tinggi.

```
X_test = X_test.reset_index(drop=True)

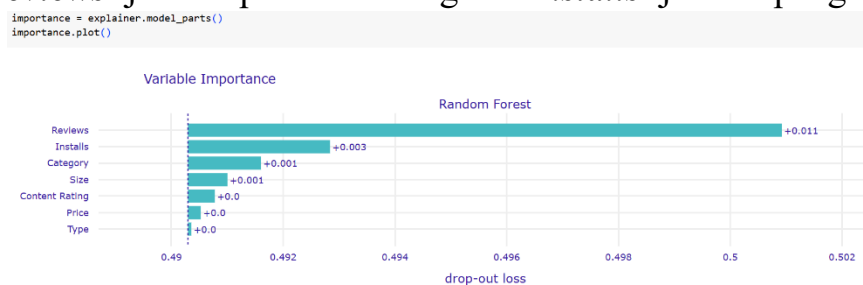
explainer = dx.Explainer(best_model_rf, X_test, y_test.astype(int), label="Random Forest")

Preparation of a new explainer is initiated

-> data : 2808 rows 7 cols
-> target variable : Parameter 'y' was a pandas.Series. Converted to a numpy.ndarray.
-> target variable : 2808 values
-> model_class : sklearn.ensemble._forest.RandomForestClassifier (default)
-> label : Random Forest
-> predict function : <function yhat_proba_default at 0x7ed716f7e0e0> will be used (default)
-> predict function : Accepts pandas.DataFrame and numpy.ndarray.
-> predicted values : min = 0.0, mean = 0.202, max = 0.869
-> model type : classification will be used (default)
-> residual function : difference between y and yhat (default)
-> residuals : min = 0.524, mean = 2.57, max = 3.0
-> model_info : package sklearn

A new explainer has been created!
```

Fitur yang paling mempengaruhi hasil prediksi model berupa *Rating* aplikasi adalah *Reviews*/ jumlah pemberi rating dan *Installs*/ jumlah pengunduh.



Garis yang menaik menandakan semakin meningkat nilai dari suatu fitur, semakin naik juga nilai prediksinya (berbanding lurus). Begitu sebaliknya. Jika garis lurus (tidak menaik maupun menurun), artinya perubahan nilai pada fitur tidak terlalu mempengaruhi nilai prediksi.



Tantangan yang Dihadapi

Rendahnya nilai akurasi model yang dibangun bahkan untuk *best model* hasil *hyperparameter tuning* disebabkan oleh beberapa faktor berikut:

1. *Imbalanced data*

Jumlah data untuk tiap-tiap kelas (*binning*) memiliki jarak yang terlalu jauh dari range 7000-an data untuk label 3 (rating tinggi) hingga 56 data untuk label 1 (rating rendah). Hal ini tentu saja membuat model lebih banyak belajar dari label 3 dibandingkan 1 karena tidak banyak informasi yang bisa yang diperoleh dari data-data berlabel 1 tersebut.

	count
Class	
3	7363
2	1941
1	56

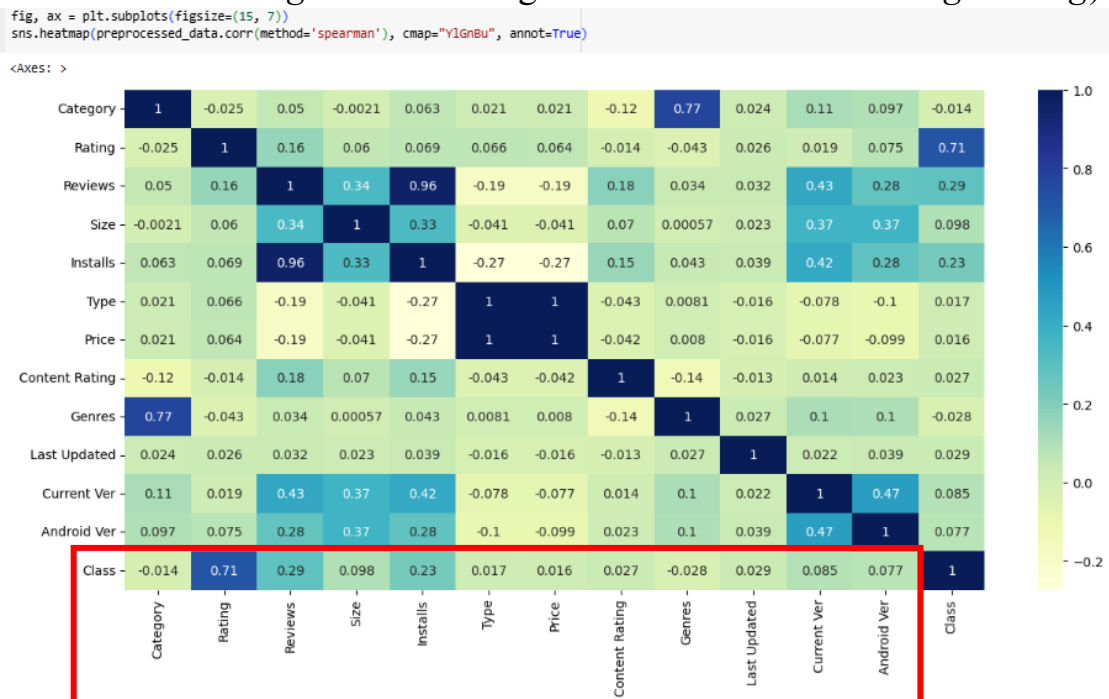
dtype: int64

Tapi mengapa tidak menyeimbangkan jumlah datanya saja saat pembagian kelas? Kami sudah mencoba untuk menyamakan jumlah data per kelas dengan pembagian: rating 1 s/d 3.5 untuk label 1, rating 3.6 s/d 4.5 untuk label 2, dan sisanya untuk label 3. Dan hasilnya, akurasi model malah menurun signifikan menjadi 40% karena pengelompokkan dilakukan dalam *range* yang kurang tepat/ sesuai.

Walaupun terdapat perbedaan jumlah data yang signifikan, model masih dapat mencapai akurasi 79% sehingga kami mempertahankan pembagian sesuai yang ditunjukkan dalam *code*.

2. Lemahnya korelasi antar variabel fitur

Sesuai yang ditampilkan dalam *heatmap*, hubungan antar variabel-variabel independen dengan variabel dependen (*class*) sangat lemah (abaikan variabel *Rating* yang memiliki nilai korelasi tinggi dengan *Class* dikarenakan *binning Class* memang dilakukan berdasarkan *range Rating*).



Kesimpulan dan Potensi Pengembangan Model

Dari keempat model yang dipilih, kami memilih model Random Forest sebagai model terbaik dari segi akurasi. Namun, untuk pengembangan model hingga implementasi ke dunia nyata sepertinya belum berpotensi untuk dilakukan karena melihat rendahnya akurasi model dalam memprediksi berdasarkan dataset GooglePlayStore yang digunakan, serta fitur-fitur yang ada didalam dataset juga masih kurang dan belum cocok/ sesuai untuk digunakan sebagai variabel independen untuk memprediksi keberhasilan aplikasi *mobile*.

Link PPT

[Akses slide presentasi GCD - Classification disini](#)

Link Notebook

[Akses notebook GCD - Classification disini](#)