KLASIFIKASI

Oleh: Tim GCD

211110347 - Cindy Sintiya
211111930 - David Bate'e
211110948 - Grace Helena Hutagaol

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

DESKRIPSI DATASET

Sumber Dataset:

kaggle.com/datasets/zeshanali/googleplaystore

01. Ukuran Dataset

```
raw_data.shape
```

```
(10841, 13)
```

02. Preview Dataset

```
raw_data = pd.read_csv("/content/googleplaystore.csv")
raw_data.sample(frac = 1).head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
10636	FRONTLINE COMMANDO	GAME	4.4	1351833	12M	10,000,000+	Free	0	Teen	Action	28-Oct-13	3.0.3	2.1 and up
721	Japanese / English one-shop search dictionary ...	EDUCATION	3.9	61	18M	50,000+	Free	0	Mature 17+	Education	28-May-18	1.4.2	5.0 and up
7598	Ben 10: Alien Experience	FAMILY	4.2	82005	45M	10,000,000+	Free	0	Everyone 10+	Action;Action & Adventure	24-Oct-17	1.2.1	4.4 and up
5930	Let's Learn Alif Ba Ta	FAMILY	4.6	32	60M	10,000+	Free	0	Everyone	Education	29-Nov-17	6	4.0.3 and up
10222	Check Your Visitors on FB ?	SOCIAL	3.6	40	1.5M	5,000+	Free	0	Everyone	Social	17-Feb-17	4.4	4.2 and up

FITUR-FITUR DATASET

Mentah

```
raw_data.info()
```

```
<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
```

PREPROCESSING DATASETS

1. Remove 'NaN' value

```
preprocessed_data = raw_data.dropna()
preprocessed_data.isna().sum()
```

	0
App	0
Category	0
Rating	0
Reviews	0
Size	0
Installs	0
Type	0
Price	0
Content Rating	0
Genres	0
Last Updated	0
Current Ver	0
Android Ver	0

2. Fixing data type

```
# 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
```

more preprocessing next ----->

PREPROCESSING DATASETS

3. Class Binning

```
# 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()
```

	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

Class	
3	7363
2	1941
1	56

more preprocessing next ----->

PREPROCESSING DATASETS

4. Feature Encoding

```
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

more preprocessing next ----->

PREPROCESSING DATASETS

5. Cek Tipe Data & Label Encode

```
# display encoded label n value

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

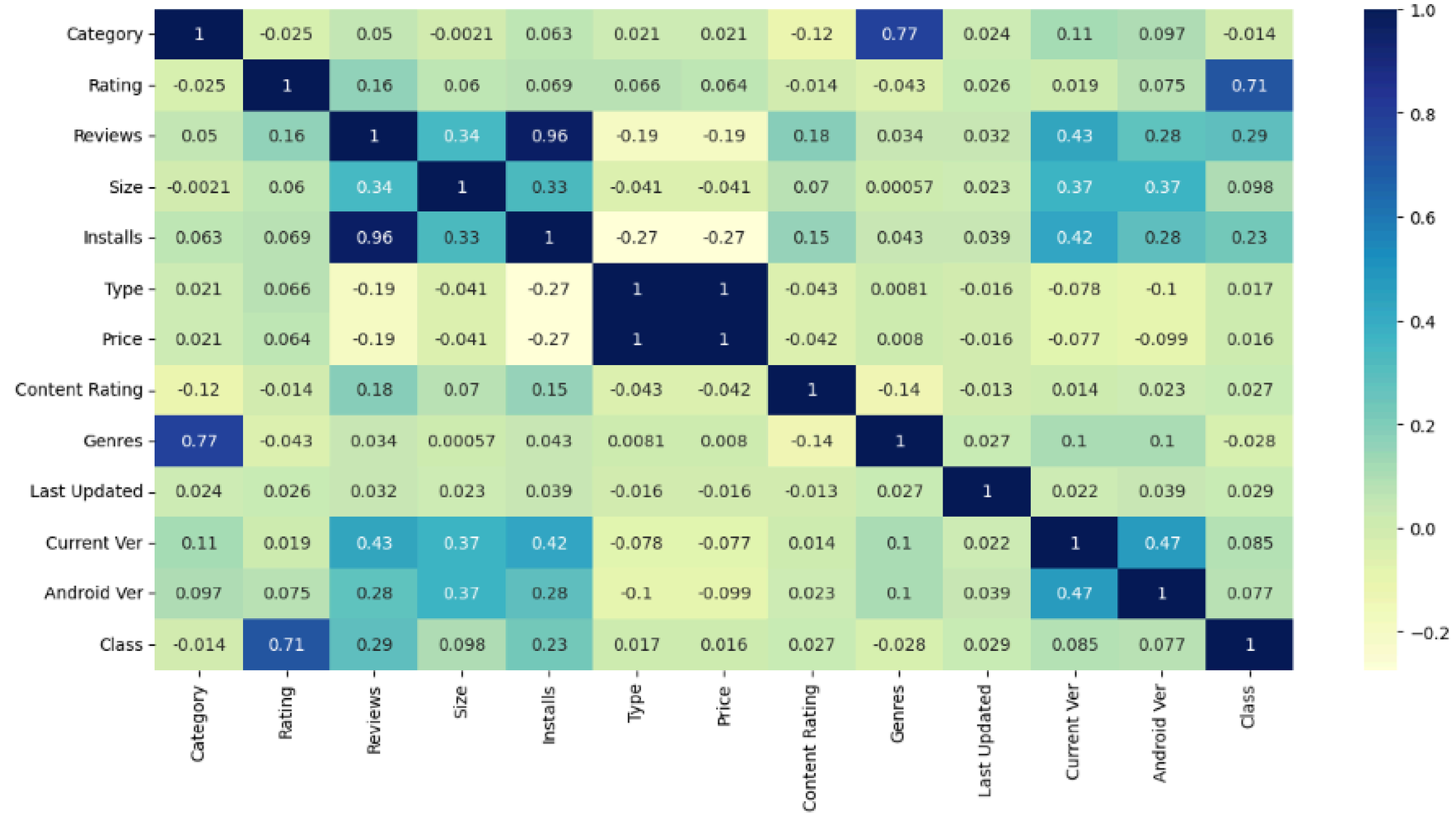
```
# cek kembali tipe data per kolom
preprocessed_data.info()
```

```
<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
```

Korelasi Antar Variabel (numerik)

```
fig, ax = plt.subplots(figsize=(15, 7))
sns.heatmap(preprocessed_data.corr(method='spearman'), cmap="YlGnBu", annot=True)
```

```
<Axes: >
```

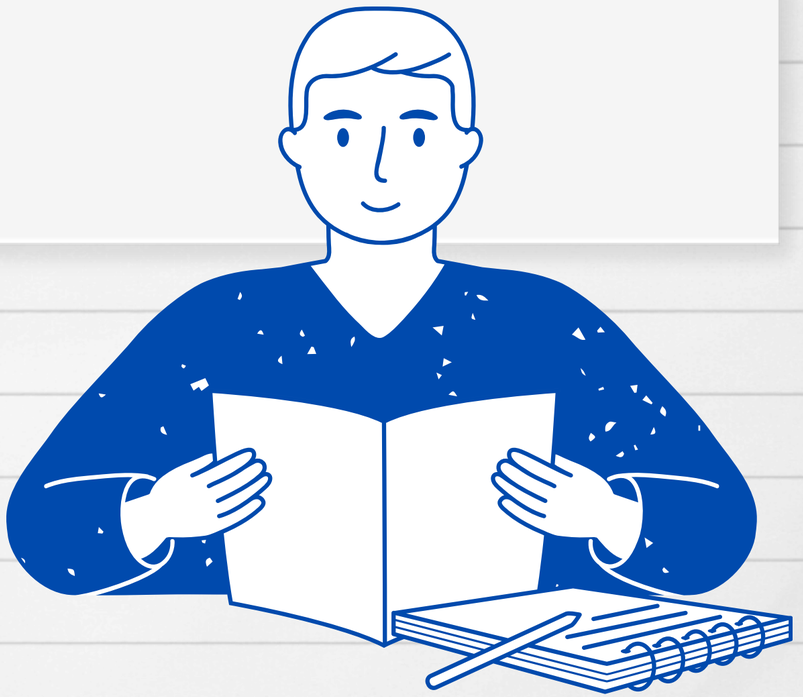


TRAIN-TEST SPLITTING

```
# 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
)
```



MODELS

Logistic Regression

```
logreg_model = LogisticRegression(max_iter=200)
logreg_model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=200)
```

Naive Bayes

```
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
```

```
GaussianNB
GaussianNB()
```

```
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
```

```
MultinomialNB
MultinomialNB()
```

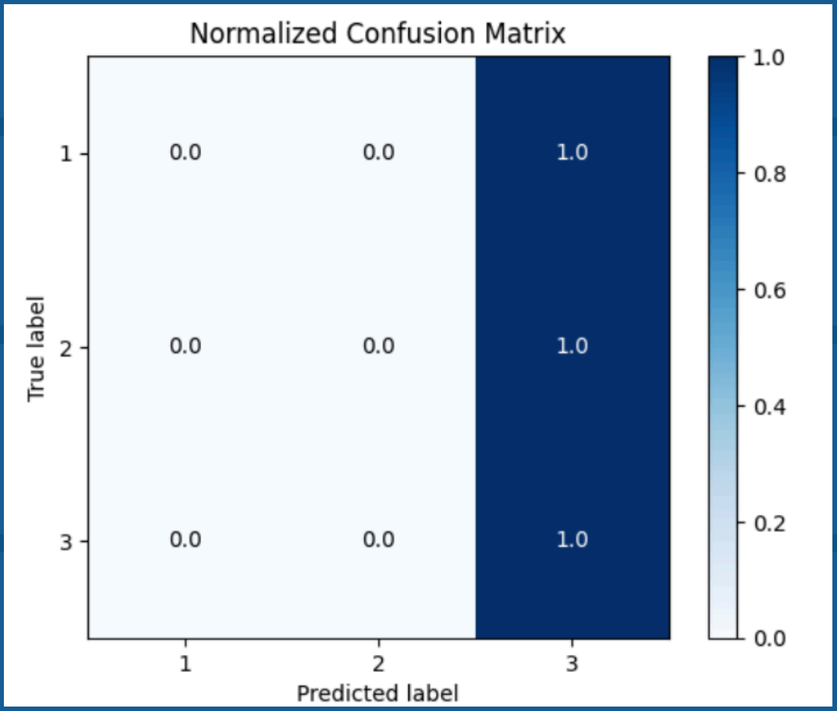
Random Forest

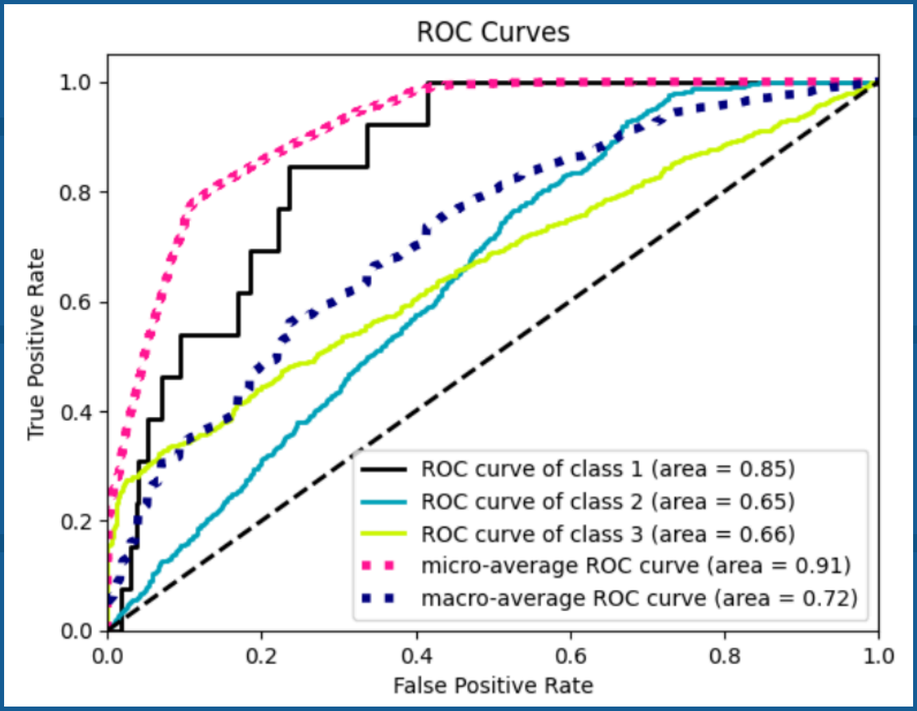
```
randfor_model = RandomForestClassifier(n_estimators=900, random_state=1000,
randfor_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(n_estimators=900, random_state=1000)
```

Evaluasi Model LOGISTIC REGRESSION

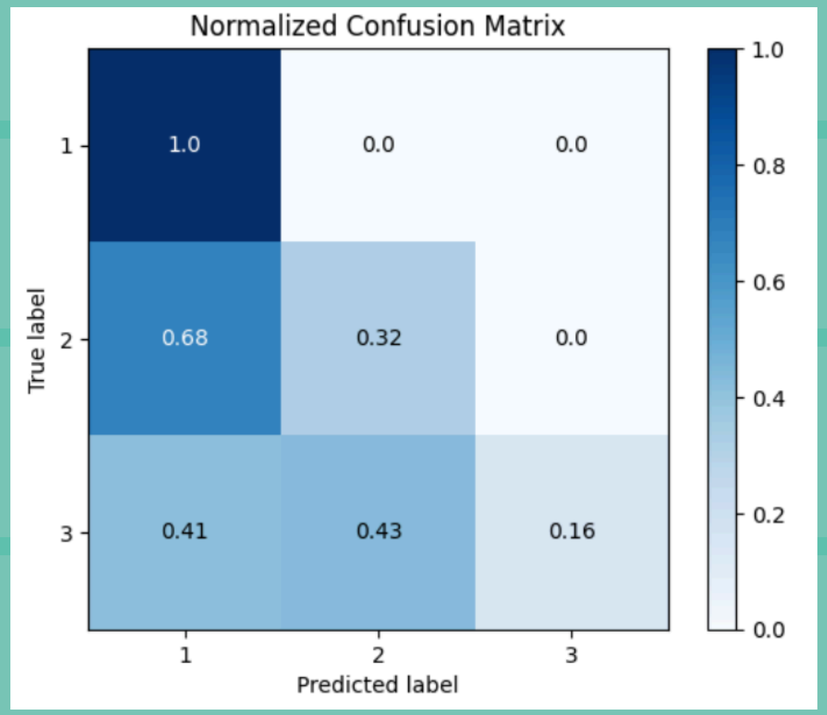
	1	2	3	accuracy	macro avg	weighted avg
precision	0.0	1.000000	0.781540	0.781695	0.593847	0.824756
recall	0.0	0.003322	1.000000	0.781695	0.334441	0.781695
f1-score	0.0	0.006623	0.877375	0.781695	0.294666	0.686635
support	13.0	602.000000	2193.000000	0.781695	2808.000000	2808.000000

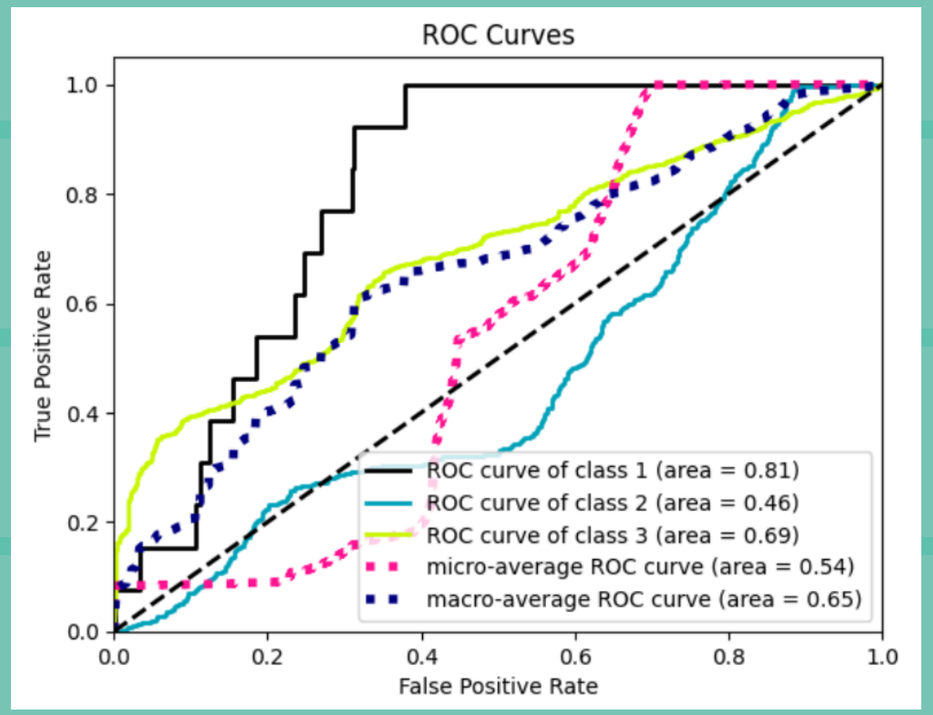




Evaluasi Model NAIVE BAYES (Gaussian)

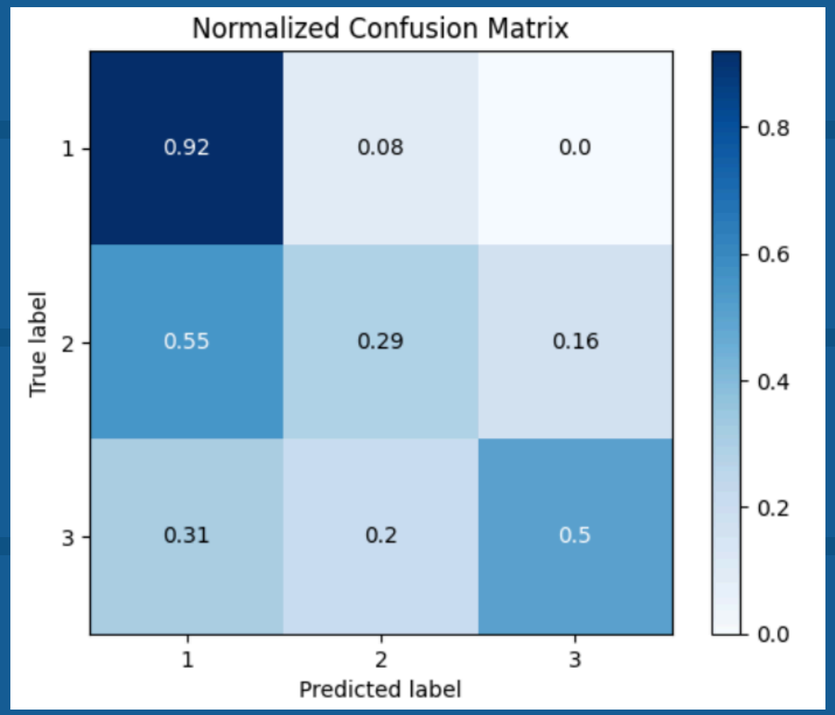
	1	2	3	accuracy	macro avg	weighted avg
precision	0.009878	0.168421	0.991477	0.197293	0.389926	0.810480
recall	1.000000	0.318937	0.159143	0.197293	0.492693	0.197293
f1-score	0.019564	0.220436	0.274263	0.197293	0.171421	0.261544
support	13.000000	602.000000	2193.000000	0.197293	2808.000000	2808.000000

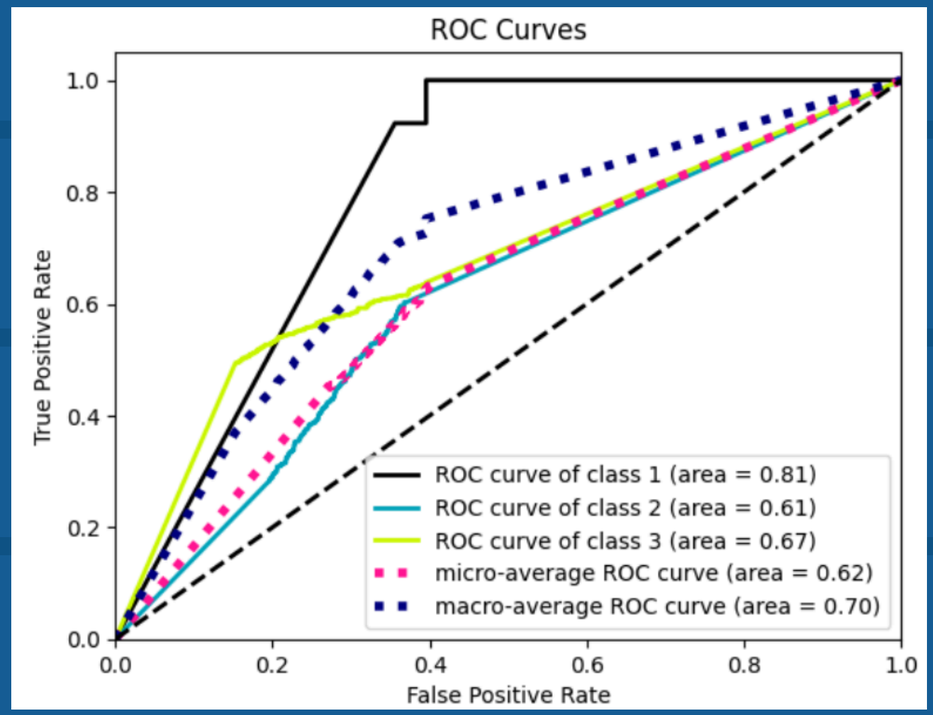




Evaluasi Model NAIVE BAYES (Multinomial)

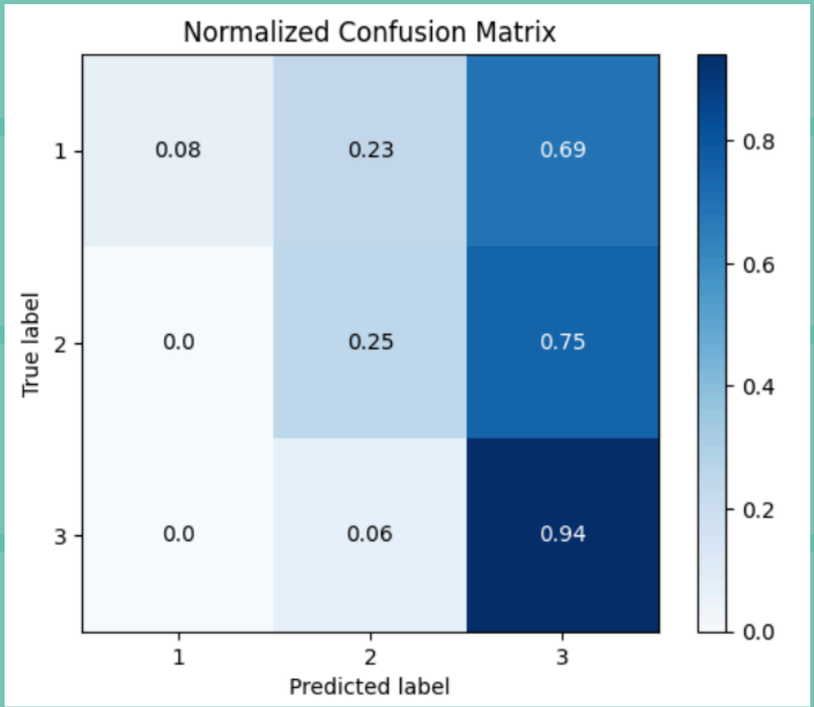
	1	2	3	accuracy	macro avg	weighted avg
precision	0.011799	0.285714	0.918782	0.452991	0.405432	0.778861
recall	0.923077	0.289037	0.495212	0.452991	0.569109	0.452991
f1-score	0.023301	0.287366	0.643556	0.452991	0.318074	0.564321
support	13.000000	602.000000	2193.000000	0.452991	2808.000000	2808.000000

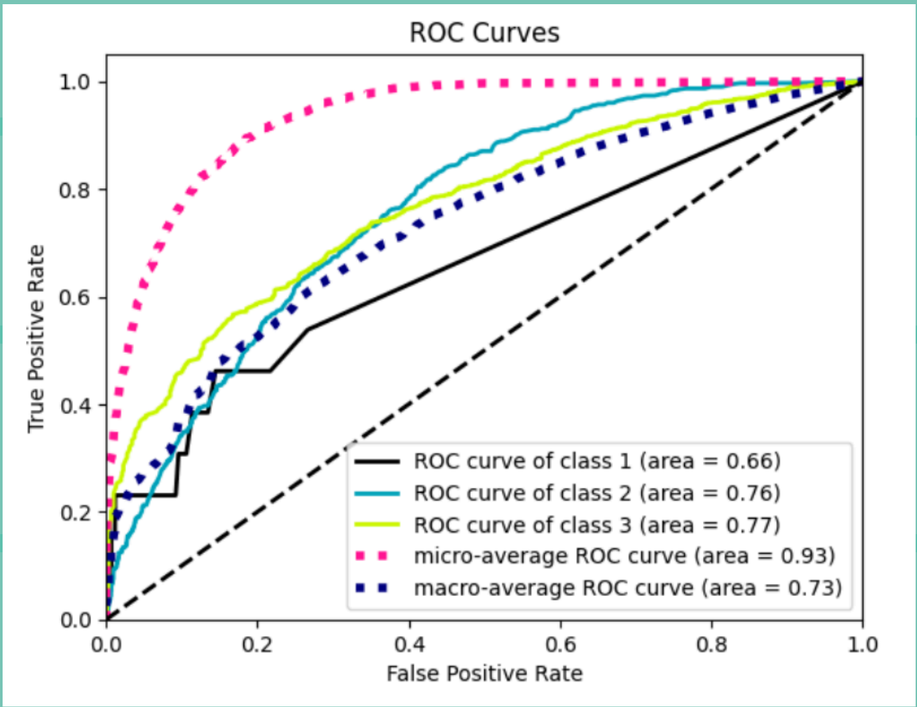




Evaluasi Model RANDOM FOREST

	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





78,56%

BEST MODEL --> RANDOM FOREST

Next for Hyperparameter Tuning --->

HYPER-PARAMETER TUNING

Choosing Random Forest as the best model with accuracy 78,56% from the other two (Naive Bayes 19,73% and Logistic Regression 78,17%)

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

# mendefinisikan parmeter 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)
```

EVALUASI MODEL: AKURASI

```
# Evaluasi model random forest hasil hyperparameter tuning
pd.DataFrame(metrics.classification_report(y_test, best_model_rf_result, target_names = classes, output_dict=True))
```

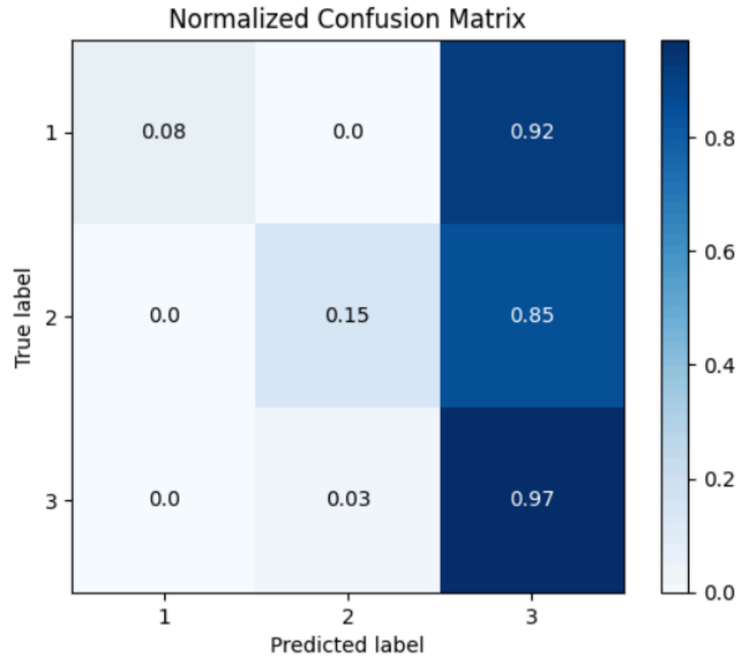
	1	2	3	accuracy	macro avg	weighted avg
precision	1.000000	0.590909	0.802865	0.791311	0.797925	0.758337
recall	0.076923	0.151163	0.971272	0.791311	0.399786	0.791311
f1-score	0.142857	0.240741	0.879076	0.791311	0.420891	0.738816
support	13.000000	602.000000	2193.000000	0.791311	2808.000000	2808.000000

79,13%

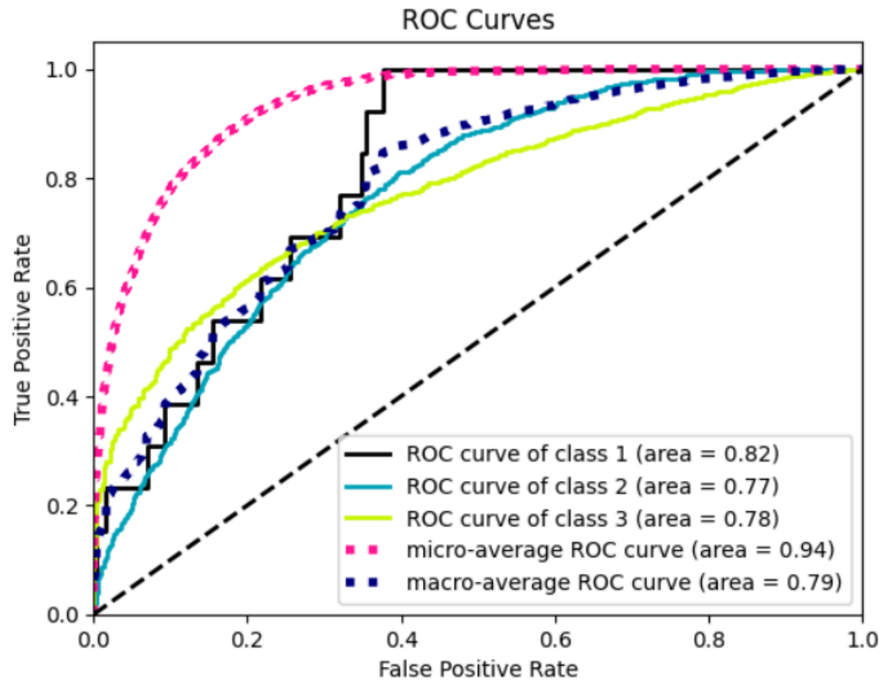
EVALUASI MODEL: CONFUSION MATRIX & ROC CURVES

```
# evaluasi model random forest
skplt.metrics.plot_confusion_matrix(y_test, best_model_rf_result, normalize=True)
```

```
<Axes: title={'center': 'Normalized Confusion Matrix'}, xlabel='Predicted label', ylabel='True label'>
```




```
# Evaluasi model random forest
skplt.metrics.plot_roc(y_test, best_model_rf_proba);
```



INTERPRETASI MODEL

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

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

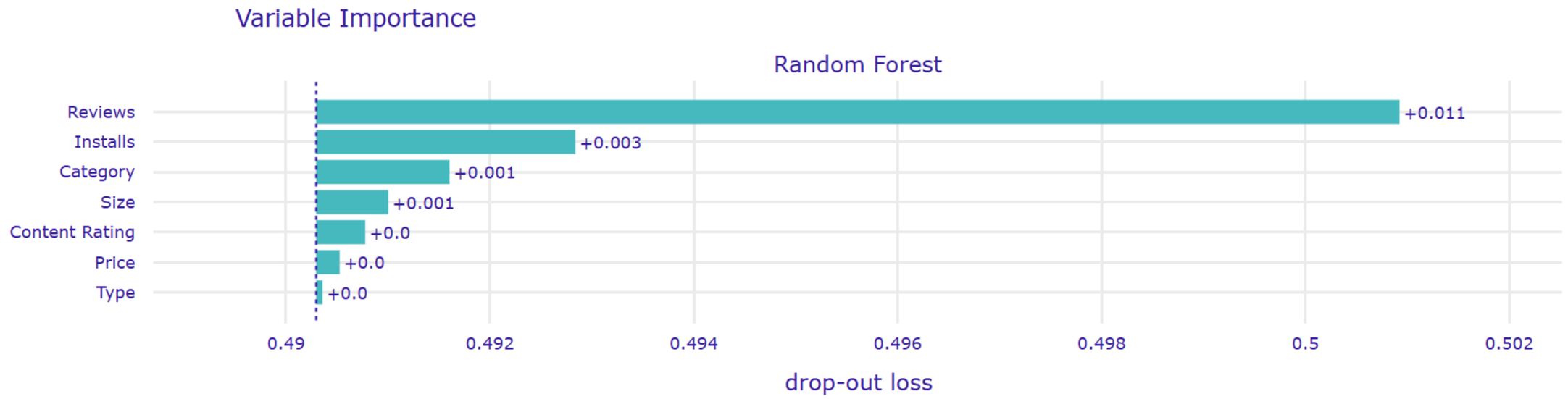
```
Preparation of a new explainer is initiated

  -> 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!
```

INTERPRETASI MODEL

```
# Feature Importance
importance = explainer.model_parts()
importance.plot()
```



INTERPRETASI MODEL

```
# Partial Dependence Plot (PDP) for the first feature
pdp = explainer.model_profile()
pdp.plot()
```

Kesimpulan

1. Imbalanced Data Label

```
# Count the number of occurrences of each class

class_counts = preprocessed_data['Class'].value_counts()
classes = sorted(class_counts.keys())
class_counts
```

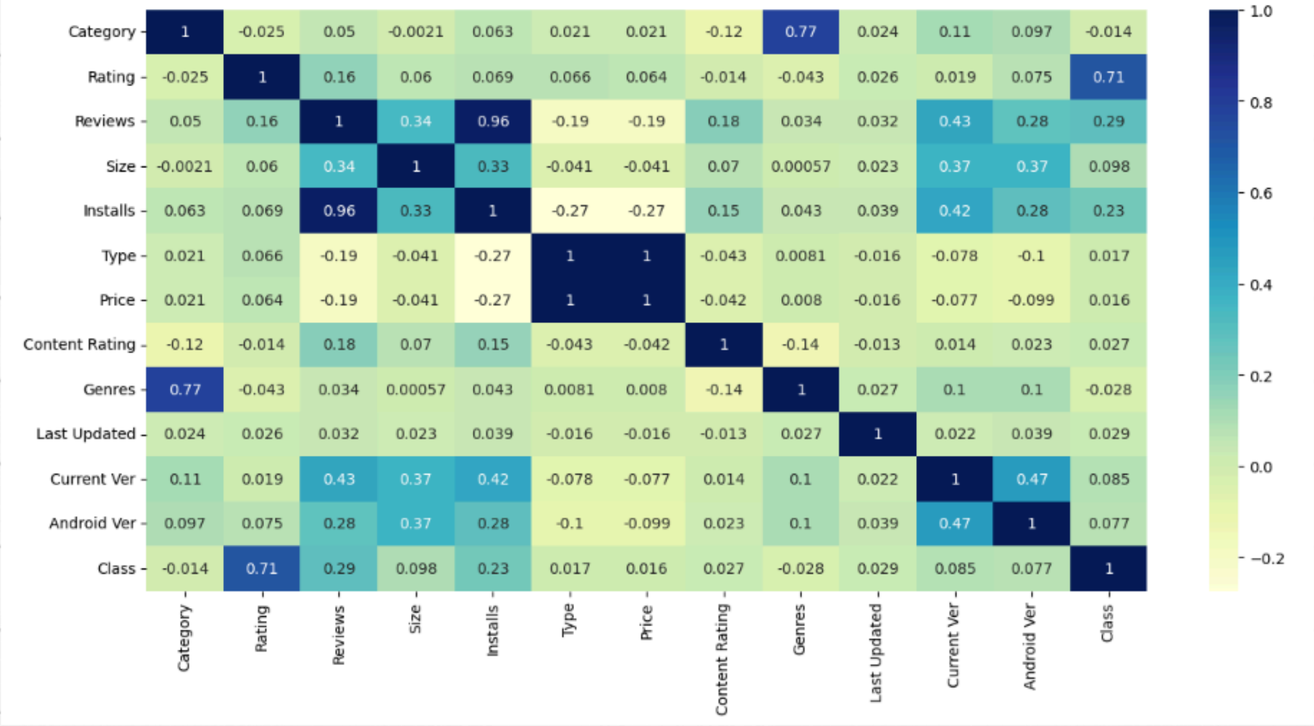
Class	count
3	7363
2	1941
1	56

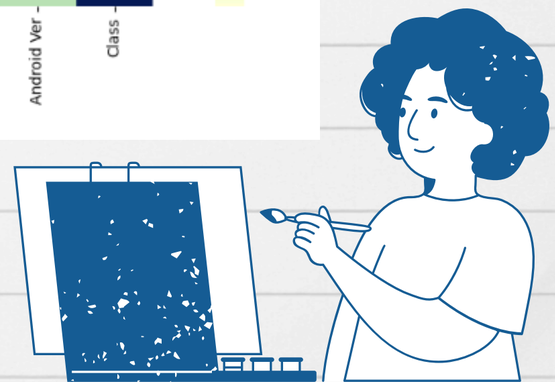
dtype: int64

2. Weak Correlation between X (features) n Y (class)

```
fig, ax = plt.subplots(figsize=(15, 7))
sns.heatmap(preprocessed_data.corr(method='spearman'), cmap="YlGnBu", annot=True)
```

<Axes: >





DONE!

Akses Coding Disini

- Tim GCD -