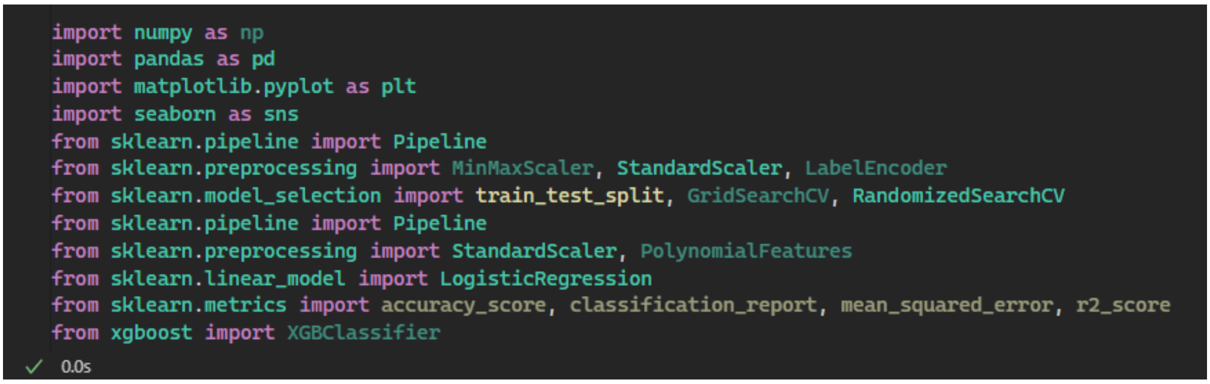
Nama : Hero Kartiko

NIM : 1103210205

UTS MACHINE LEARNING

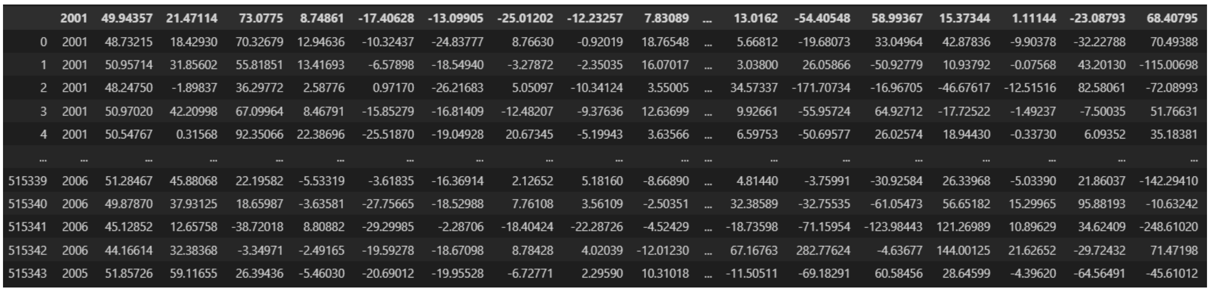
1. Dataset Regressi UTS Telkom

- Library yang akan digunakan di pengolahan regresi.



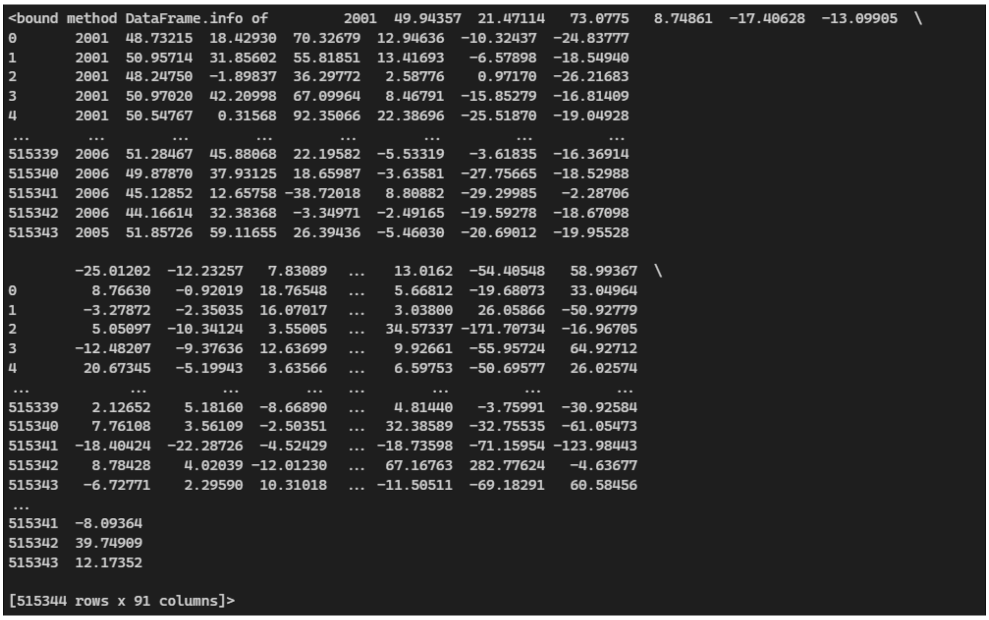
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, mean_squared_error, r2_score
from xgboost import XGBClassifier
```

- Membaca dataset dengan library pandas pd.read_csv(nama_dataset)



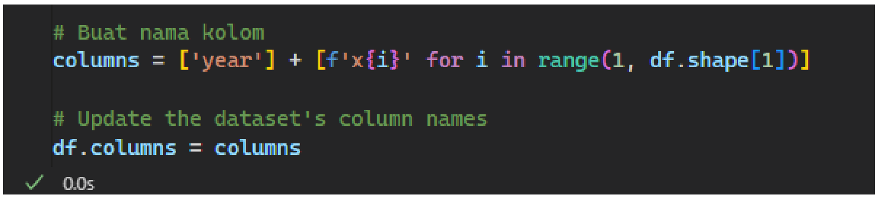
	2001	49.94357	21.47114	73.0775	8.74861	-17.40628	-13.09905	-25.01202	-12.23257	7.83089	...	13.0162	-54.40548	58.99367	15.37344	1.11144	-23.08793	68.40795
0	2001	48.73215	18.42930	70.32679	12.94636	-10.32437	-24.83777	8.76630	-0.92019	18.76548	...	5.66812	-19.68073	33.04964	42.87836	-9.90378	-32.22788	70.49388
1	2001	50.95714	31.85602	55.81851	13.41693	-6.57898	-18.54940	-3.27872	-2.35035	16.07017	...	3.03800	26.05866	-50.92779	10.93792	-0.07568	43.20130	-115.00698
2	2001	48.24750	-1.89837	36.29772	2.58776	0.97170	-26.21683	5.05097	-10.34124	3.55005	...	34.57337	-171.70734	-16.96705	-46.67617	-12.51516	82.58061	-72.08993
3	2001	50.97020	42.20998	67.09964	8.46791	-15.85279	-16.81409	-12.48207	-9.37636	12.63699	...	9.92661	-55.95724	64.92712	-17.72522	-1.49237	-7.50035	51.76631
4	2001	50.54767	0.31568	92.35066	22.38696	-25.51870	-19.04928	20.67345	-5.19943	3.63566	...	6.59753	-50.69577	26.02574	18.94430	-0.33730	6.09352	35.18381
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
515339	2006	51.28467	45.88068	22.19582	-5.53319	-3.61835	-16.36914	2.12652	5.18160	-8.66890	...	4.81440	-3.75991	-30.92584	26.33968	-5.03390	21.86037	-142.29410
515340	2006	49.87870	37.93125	18.65987	-3.63581	-27.75665	-18.52988	7.76108	3.56109	-2.50351	...	32.38589	-32.75535	-61.05473	56.65182	15.29965	95.88193	-10.63242
515341	2006	45.12852	12.65758	-38.72018	8.80882	-29.29985	-2.28706	-18.40424	-22.28726	-4.52429	...	-18.73598	-71.15954	-123.98443	121.26989	10.89629	34.62409	-248.61020
515342	2006	44.16614	32.38368	-3.34971	-2.49165	-19.59278	-18.67098	8.78428	4.02039	-12.01230	...	67.16763	282.77624	-4.63677	144.00125	21.62652	-29.72432	71.47198
515343	2005	51.85726	59.11655	26.39436	-5.46030	-20.69012	-19.95528	-6.72771	2.29590	10.31018	...	-11.50511	-69.18291	60.58456	28.64599	-4.39620	-64.56491	-45.61012

- Menampilkan info dataset yang memiliki 515344 baris dengan 91 kolom.



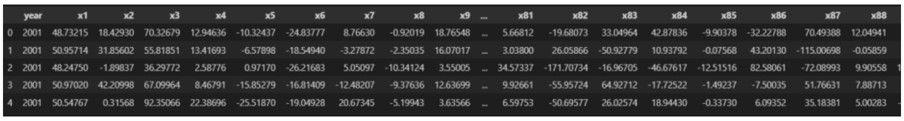
```
<bound method DataFrame.info of         2001  49.94357  21.47114   73.0775   8.74861  -17.40628  -13.09905  \
0       2001  48.73215  18.42930  70.32679  12.94636  -10.32437  -24.83777
1       2001  50.95714  31.85602  55.81851  13.41693   -6.57898  -18.54940
2       2001  48.24750  -1.89837  36.29772   2.58776    0.97170  -26.21683
3       2001  50.97020  42.20998  67.09964   8.46791  -15.85279  -16.81409
4       2001  50.54767   0.31568  92.35066  22.38696  -25.51870  -19.04928
 ...     ...       ...       ...       ...       ...        ...        ...
515339  2006  51.28467  45.88068  22.19582  -5.53319   -3.61835  -16.36914
515340  2006  49.87870  37.93125  18.65987  -3.63581  -27.75665  -18.52988
515341  2006  45.12852  12.65758 -38.72018   8.80882  -29.29985   -2.28706
515342  2006  44.16614  32.38368  -3.34971  -2.49165  -19.59278  -18.67098
515343  2005  51.85726  59.11655  26.39436  -5.46030  -20.69012  -19.95528

        -25.01202  -12.23257   7.83089  ...   13.0162  -54.40548   58.99367  \
0         8.76630   -0.92019  18.76548  ...   5.66812  -19.68073   33.04964
1        -3.27872   -2.35035  16.07017  ...   3.03800   26.05866  -50.92779
2         5.05097  -10.34124   3.55005  ...  34.57337 -171.70734  -16.96705
3       -12.48207   -9.37636  12.63699  ...   9.92661  -55.95724   64.92712
4        20.67345   -5.19943   3.63566  ...   6.59753  -50.69577   26.02574
 ...          ...        ...       ...  ...       ...        ...        ...
515339    2.12652    5.18160  -8.66890  ...   4.81440   -3.75991  -30.92584
515340    7.76108    3.56109  -2.50351  ...  32.38589  -32.75535  -61.05473
515341  -18.40424  -22.28726  -4.52429  ... -18.73598  -71.15954 -123.98443
515342    8.78428    4.02039 -12.01230  ...  67.16763  282.77624   -4.63677
515343   -6.72771    2.29590  10.31018  ... -11.50511  -69.18291   60.58456
 ...
515341  -8.09364
515342  39.74909
515343  12.17352

[515344 rows x 91 columns]>
```

- Memberi nama salah fitur dengan ’year’ untuk mempermudah penargetan dataset.



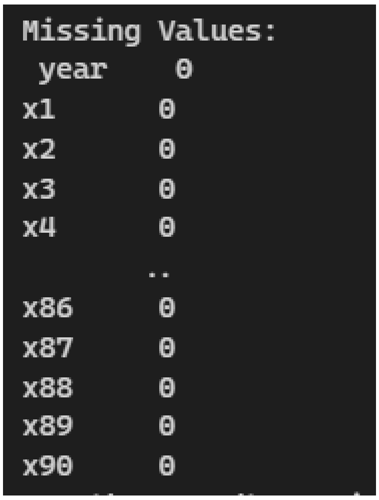
```
# Buat nama kolom
columns = ['year'] + [f'x{i}' for i in range(1, df.shape[1])]

# Update the dataset's column names
df.columns = columns
```

✓ 0.0s



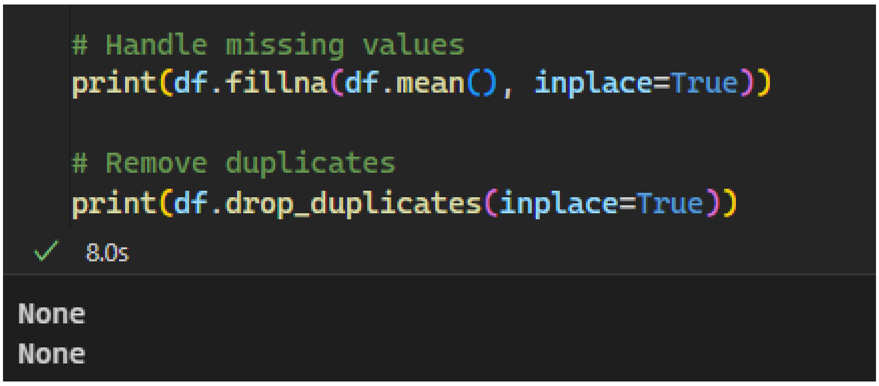
	year	x1	x2	x3	x4	x5	x6	x7	x8	x9	...	x81	x82	x83	x84	x85	x86	x87	x88
0	2001	48.73215	18.42930	70.32679	12.94636	-10.32437	-24.83777	8.76630	-0.92019	18.76548	...	5.66812	-19.68073	33.04964	42.87836	-9.90378	-32.22788	70.49388	12.04941
1	2001	50.95714	31.85602	55.81851	13.41693	-6.57898	-18.54940	-3.27872	-2.35035	16.07017	...	3.03800	26.05866	-50.92779	10.93792	-0.07568	43.20130	-115.00698	-0.05859
2	2001	48.24750	-1.89837	36.29772	2.58776	0.97170	-26.21683	5.05097	-10.34124	3.55005	...	34.57337	-171.70734	-16.96705	-46.67617	-12.51516	82.58061	-72.08993	9.90558
3	2001	50.97020	42.20998	67.09964	8.46791	-15.85279	-16.81409	-12.48207	-9.37636	12.63699	...	9.92661	-55.95724	64.92712	-17.72522	-1.49237	-7.50035	51.76631	7.88713
4	2001	50.54767	0.31568	92.35066	22.38696	-25.51870	-19.04928	20.67345	-5.19943	3.63566	...	6.59753	-50.69577	26.02574	18.94430	-0.33730	6.09352	35.18381	5.00283

- Melakukan pengecekan missing values.



```
Missing Values:
 year    0
x1       0
x2       0
x3       0
x4       0
        ..
x86      0
x87      0
x88      0
x89      0
x90      0
```

- Melakukan analisis fitur yang hilang.

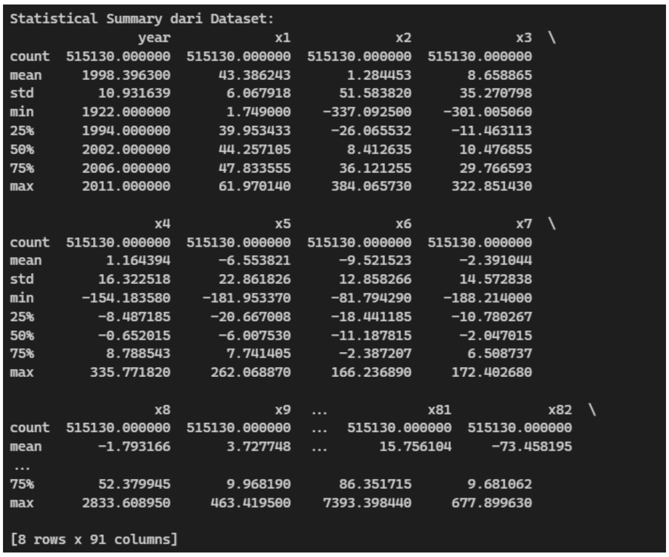


```
# Handle missing values
print(df.fillna(df.mean(), inplace=True))

# Remove duplicates
print(df.drop_duplicates(inplace=True))
```

✓ 8.0s

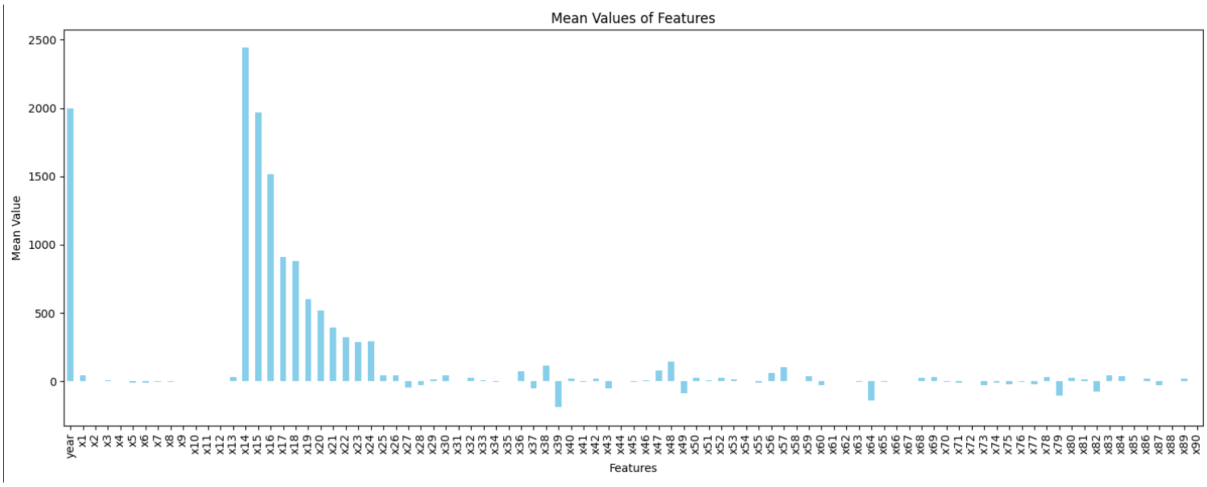
```
None
None
```

- Menampilkan Statistical Summary dari dataset.

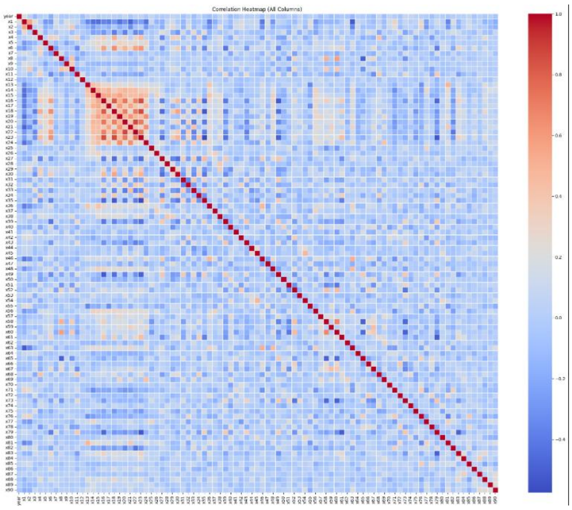


```
Statistical Summary dari Dataset:
                year             x1             x2             x3  \
count  515130.000000  515130.000000  515130.000000  515130.000000
mean     1998.396300      43.386243       1.284453       8.658865
std        10.931639       6.067918      51.583820      35.270798
min      1922.000000       1.749000    -337.092500    -301.005060
25%      1994.000000      39.953433     -26.065532     -11.463113
50%      2002.000000      44.257105       8.412635      10.476855
75%      2006.000000      47.833555      36.121255      29.766593
max      2011.000000      61.970140     384.065730     322.851430

                  x4             x5             x6             x7  \
count  515130.000000  515130.000000  515130.000000  515130.000000
mean        1.164394      -6.553821      -9.521523      -2.391044
std        16.322518      22.861826      12.858266      14.572838
min      -154.183580    -181.953370     -81.794290    -188.214000
25%        -8.487185     -20.667008     -18.441185     -10.780267
50%        -0.652015      -6.007530     -11.187815      -2.047015
75%         8.788543       7.741405      -2.387207       6.508737
max       335.771820     262.068870     166.236890     172.402680

                  x8             x9  ...            x81            x82  \
count  515130.000000  515130.000000  ...  515130.000000  515130.000000
mean       -1.793166       3.727748  ...      15.756104     -73.458195
...
75%        52.379945       9.968190      86.351715       9.681062
max      2833.608950     463.419500    7393.398440     677.899630

[8 rows x 91 columns]
```

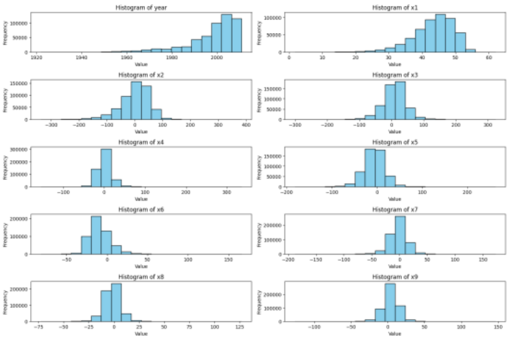
- Visualisasi nilai Mean pada dataset.



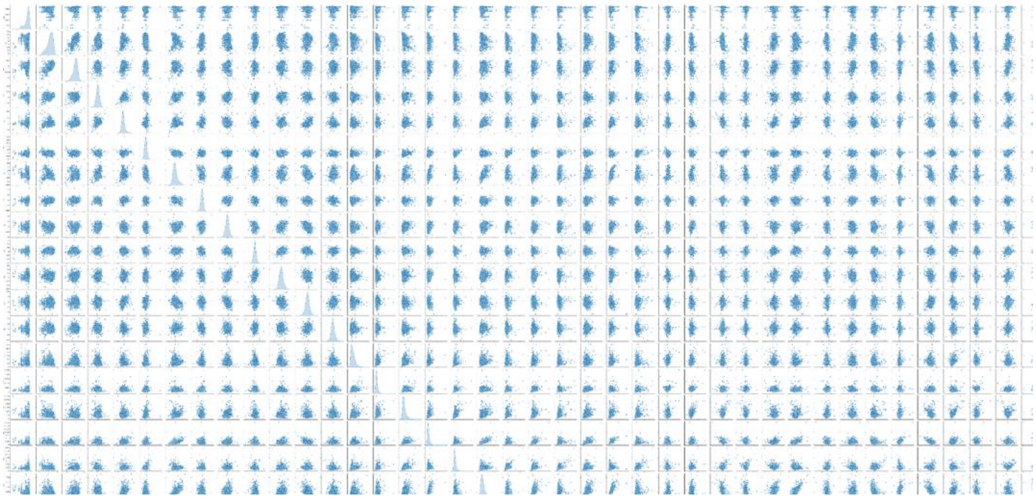
- Visualisasi heatmap dari dataset.



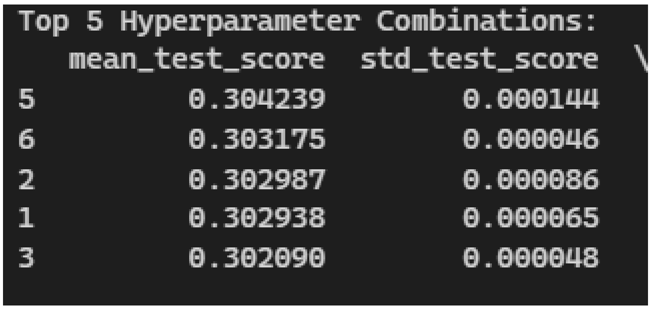
- Visualisasi histogram dari beberapa fitur.



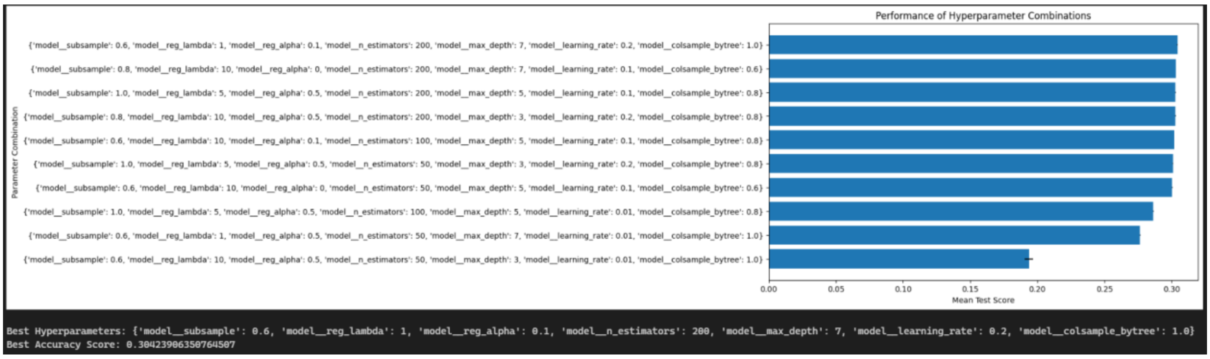
- Visualisasi Scatter Plot.



- Visualisasi output model klasifikasi dengan hyperparameter basis function.

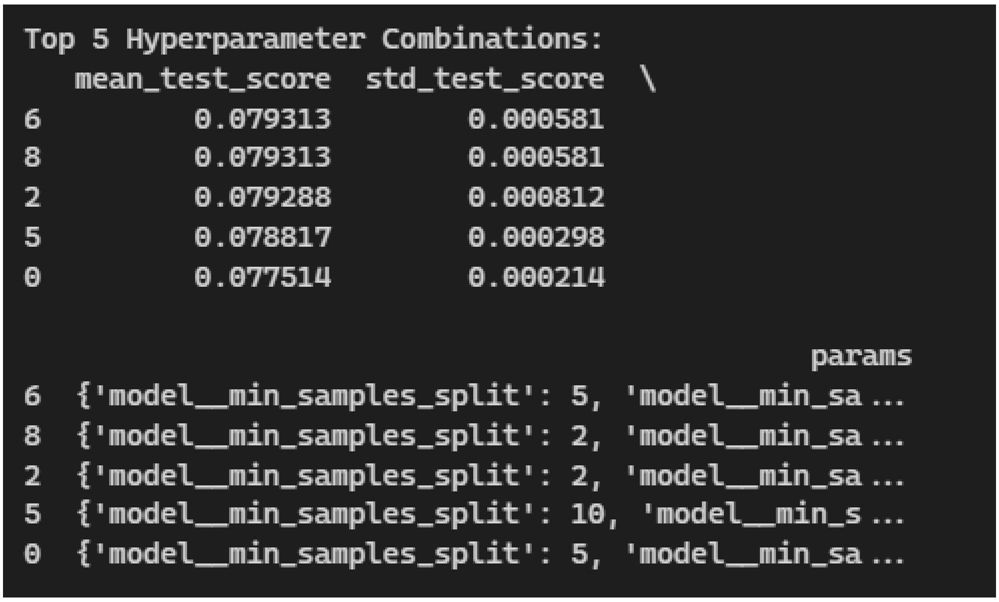


```
Top 5 Hyperparameter Combinations:
   mean_test_score  std_test_score  \
5         0.304239        0.000144
6         0.303175        0.000046
2         0.302987        0.000086
1         0.302938        0.000065
3         0.302090        0.000048
```

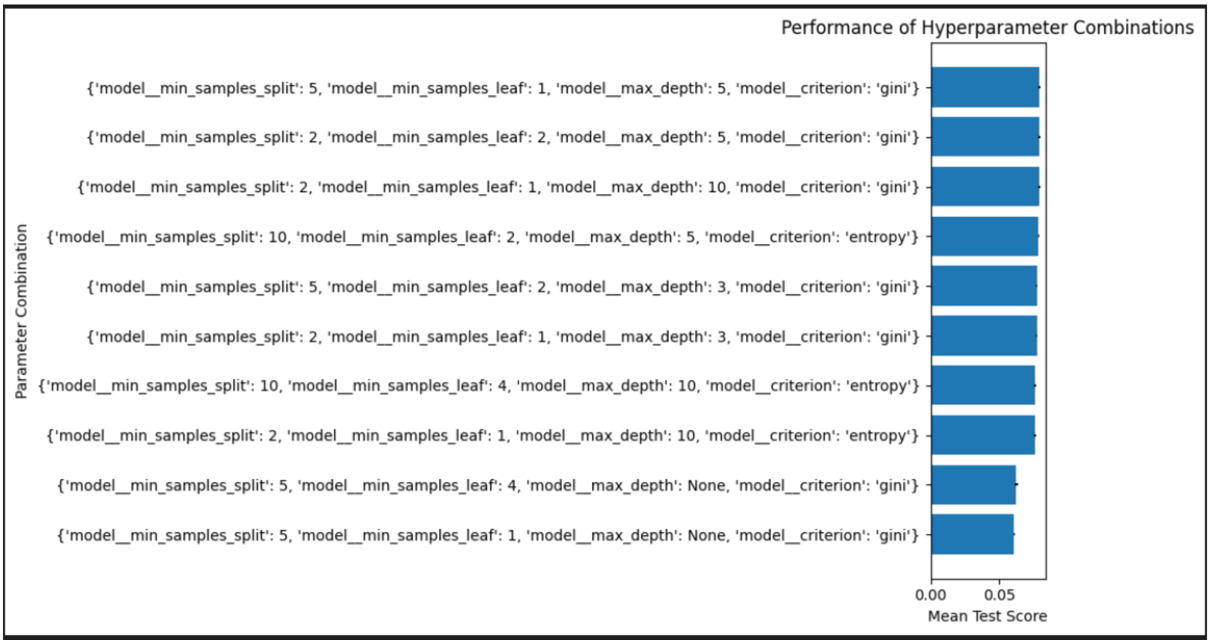

Performance of Hyperparameter Combinations
Parameter Combination
{'model__subsample': 0.6, 'model__reg_lambda': 1, 'model__reg_alpha': 0.1, 'model__n_estimators': 200, 'model__max_depth': 7, 'model__learning_rate': 0.2, 'model__colsample_bytree': 1.0}
{'model__subsample': 0.8, 'model__reg_lambda': 10, 'model__reg_alpha': 0, 'model__n_estimators': 200, 'model__max_depth': 7, 'model__learning_rate': 0.1, 'model__colsample_bytree': 0.6}
{'model__subsample': 1.0, 'model__reg_lambda': 5, 'model__reg_alpha': 0.5, 'model__n_estimators': 200, 'model__max_depth': 5, 'model__learning_rate': 0.1, 'model__colsample_bytree': 0.8}
{'model__subsample': 0.8, 'model__reg_lambda': 10, 'model__reg_alpha': 0.5, 'model__n_estimators': 200, 'model__max_depth': 3, 'model__learning_rate': 0.2, 'model__colsample_bytree': 0.8}
{'model__subsample': 0.6, 'model__reg_lambda': 10, 'model__reg_alpha': 0.1, 'model__n_estimators': 100, 'model__max_depth': 5, 'model__learning_rate': 0.1, 'model__colsample_bytree': 0.8}
{'model__subsample': 1.0, 'model__reg_lambda': 5, 'model__reg_alpha': 0.5, 'model__n_estimators': 50, 'model__max_depth': 3, 'model__learning_rate': 0.2, 'model__colsample_bytree': 0.8}
{'model__subsample': 0.6, 'model__reg_lambda': 10, 'model__reg_alpha': 0, 'model__n_estimators': 50, 'model__max_depth': 5, 'model__learning_rate': 0.1, 'model__colsample_bytree': 0.6}
{'model__subsample': 1.0, 'model__reg_lambda': 5, 'model__reg_alpha': 0.5, 'model__n_estimators': 100, 'model__max_depth': 5, 'model__learning_rate': 0.01, 'model__colsample_bytree': 0.8}
{'model__subsample': 0.6, 'model__reg_lambda': 1, 'model__reg_alpha': 0.5, 'model__n_estimators': 50, 'model__max_depth': 7, 'model__learning_rate': 0.01, 'model__colsample_bytree': 1.0}
{'model__subsample': 0.6, 'model__reg_lambda': 10, 'model__reg_alpha': 0.5, 'model__n_estimators': 50, 'model__max_depth': 3, 'model__learning_rate': 0.01, 'model__colsample_bytree': 1.0}
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Mean Test Score

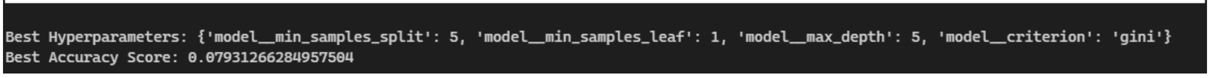
```
Best Hyperparameters: {'model__subsample': 0.6, 'model__reg_lambda': 1, 'model__reg_alpha': 0.1, 'model__n_estimators': 200, 'model__max_depth': 7, 'model__learning_rate': 0.2, 'model__colsample_bytree': 1.0}
Best Accuracy Score: 0.30423906350764507
```

- Visualisasi output model klasifikasi dengan hyperparameter decision tree.



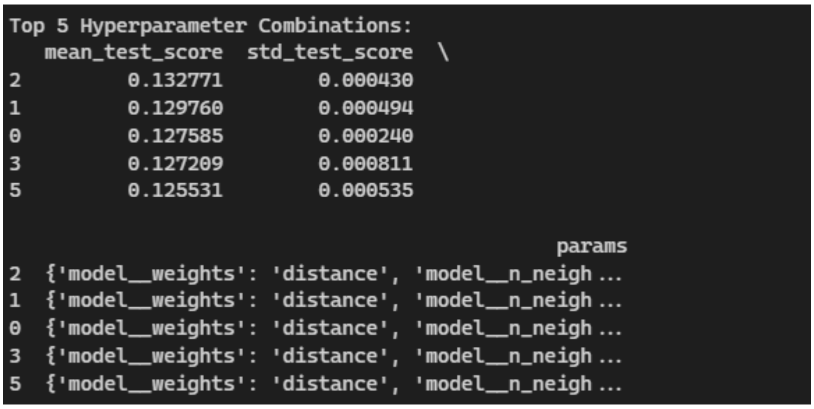
```
Top 5 Hyperparameter Combinations:
   mean_test_score  std_test_score  \
6         0.079313        0.000581
8         0.079313        0.000581
2         0.079288        0.000812
5         0.078817        0.000298
0         0.077514        0.000214

                                              params
6  {'model__min_samples_split': 5, 'model__min_sa...
8  {'model__min_samples_split': 2, 'model__min_sa...
2  {'model__min_samples_split': 2, 'model__min_sa...
5  {'model__min_samples_split': 10, 'model__min_s...
0  {'model__min_samples_split': 5, 'model__min_sa...
```



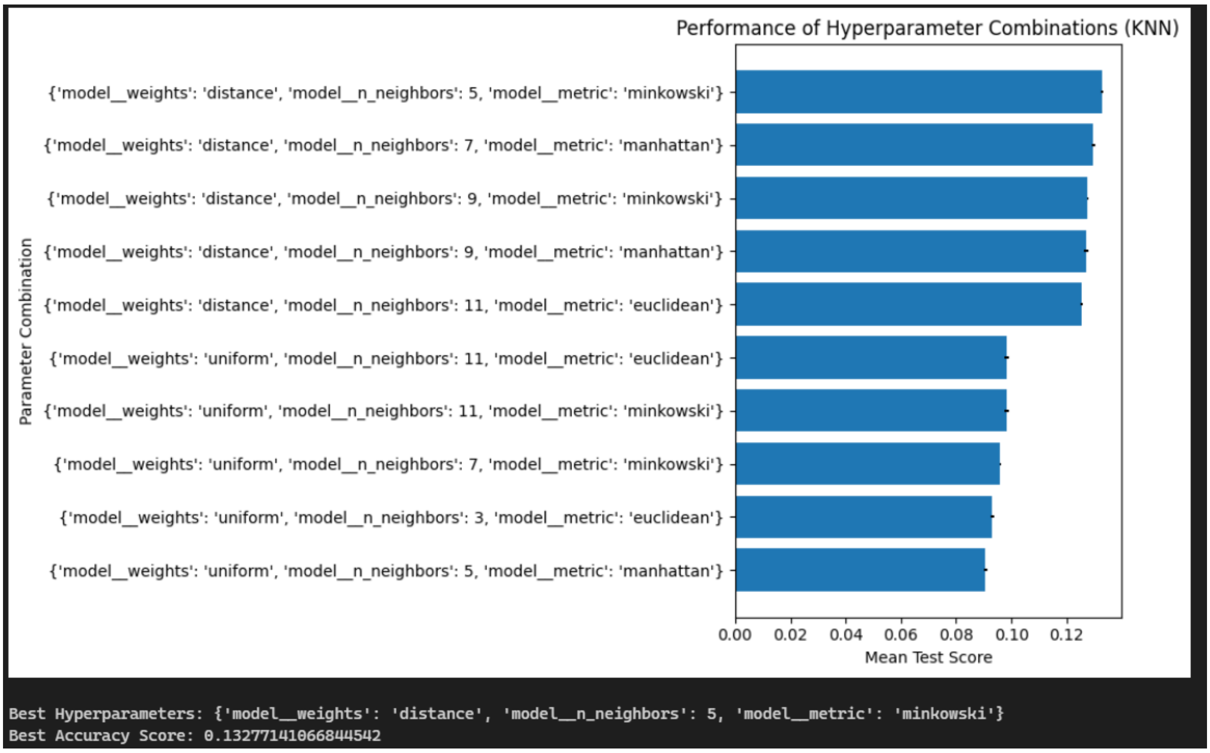


```
Best Hyperparameters: {'model__min_samples_split': 5, 'model__min_samples_leaf': 1, 'model__max_depth': 5, 'model__criterion': 'gini'}
Best Accuracy Score: 0.07931266284957504
```

- Visualisasi output model klasifikasi dengan hyperparameter KNN

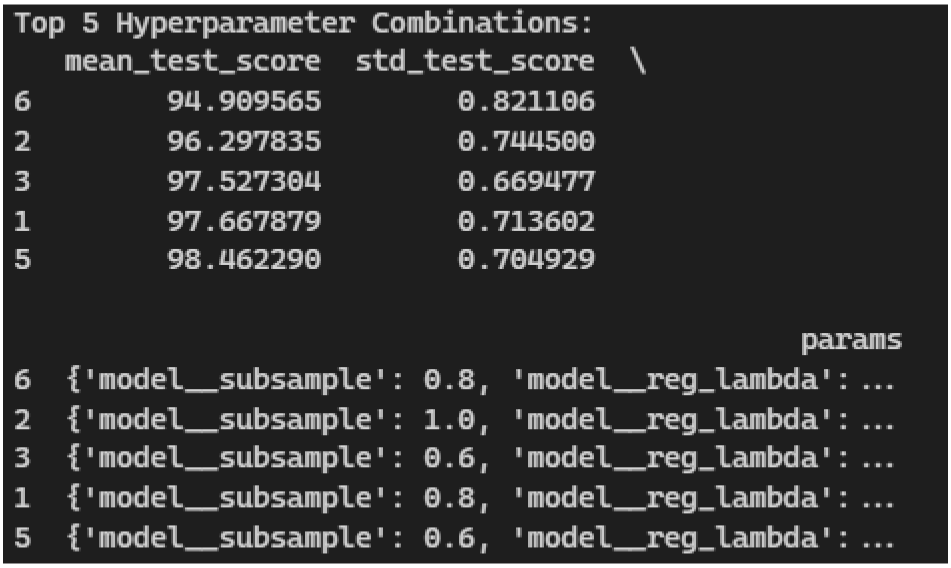


```
Top 5 Hyperparameter Combinations:
   mean_test_score  std_test_score  \
2         0.132771        0.000430
1         0.129760        0.000494
0         0.127585        0.000240
3         0.127209        0.000811
5         0.125531        0.000535

                                              params
2  {'model__weights': 'distance', 'model__n_neigh...
1  {'model__weights': 'distance', 'model__n_neigh...
0  {'model__weights': 'distance', 'model__n_neigh...
3  {'model__weights': 'distance', 'model__n_neigh...
5  {'model__weights': 'distance', 'model__n_neigh...
```

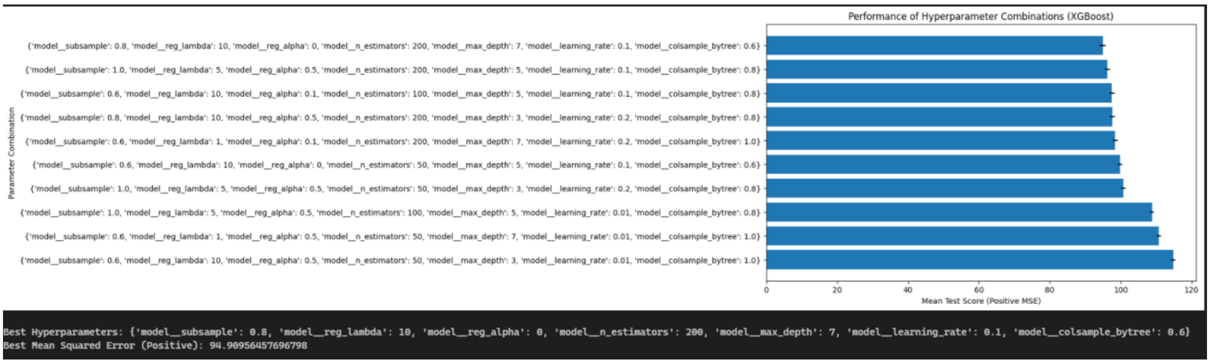


```
Best Hyperparameters: {'model__weights': 'distance', 'model__n_neighbors': 5, 'model__metric': 'minkowski'}
Best Accuracy Score: 0.13277141066844542
```

- Visualisasi output model klasifikasi dengan hyperparameter XGBoost



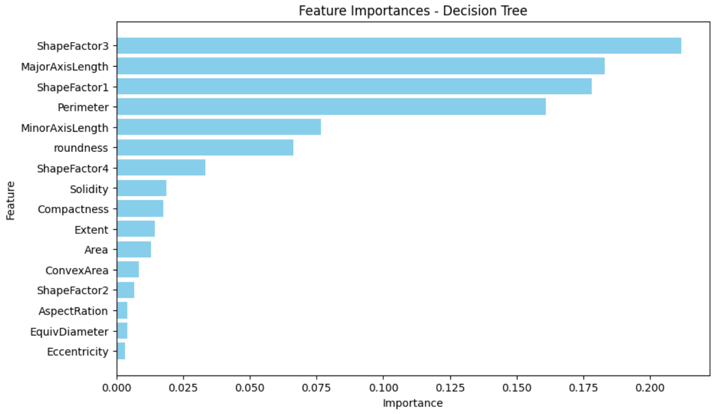
```
Top 5 Hyperparameter Combinations:
   mean_test_score  std_test_score  \
6        94.909565        0.821106
2        96.297835        0.744500
3        97.527304        0.669477
1        97.667879        0.713602
5        98.462290        0.704929

                                              params
6  {'model__subsample': 0.8, 'model__reg_lambda': ...
2  {'model__subsample': 1.0, 'model__reg_lambda': ...
3  {'model__subsample': 0.6, 'model__reg_lambda': ...
1  {'model__subsample': 0.8, 'model__reg_lambda': ...
5  {'model__subsample': 0.6, 'model__reg_lambda': ...
```



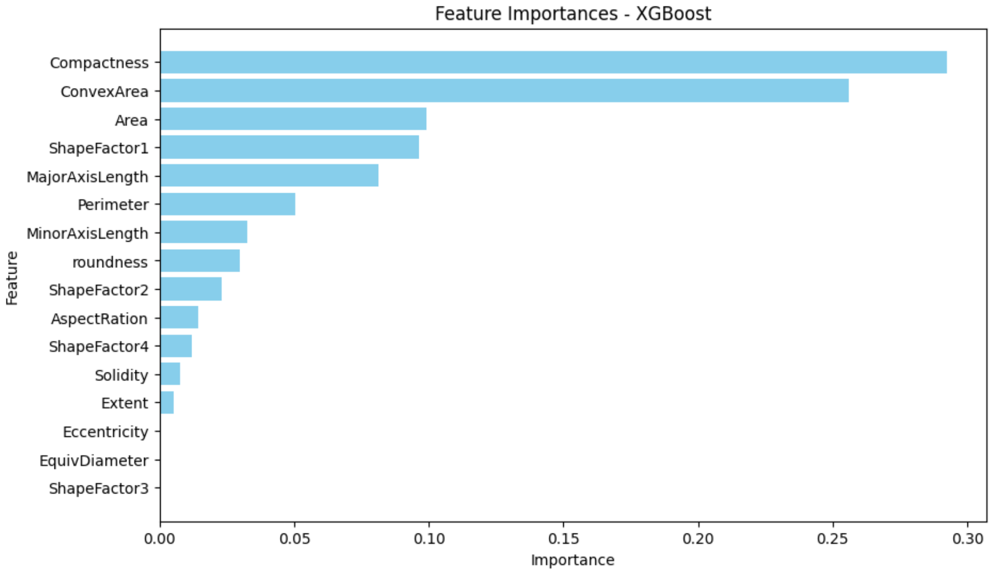
```
Best Hyperparameters: {'model__subsample': 0.8, 'model__reg_lambda': 10, 'model__reg_alpha': 0, 'model__n_estimators': 200, 'model__max_depth': 7, 'model__learning_rate': 0.1, 'model__colsample_bytree': 0.6}
Best Mean Squared Error (Positive): 94.90956457696798
```

2. Classification Dataset Dry Bean

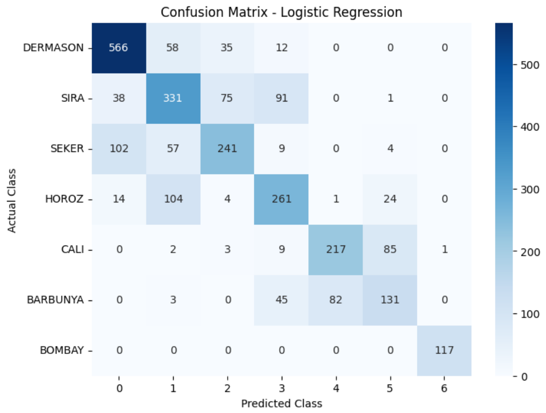
- Feautre Importance berdasarkan model decision tree

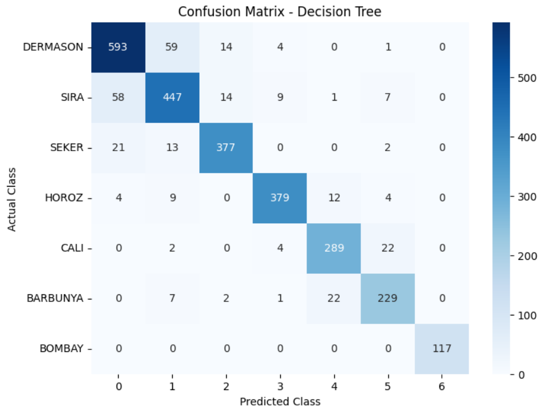


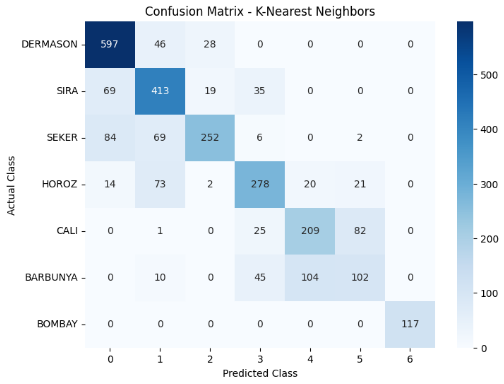
- Feautre Importance berdasarkan model XGBoost

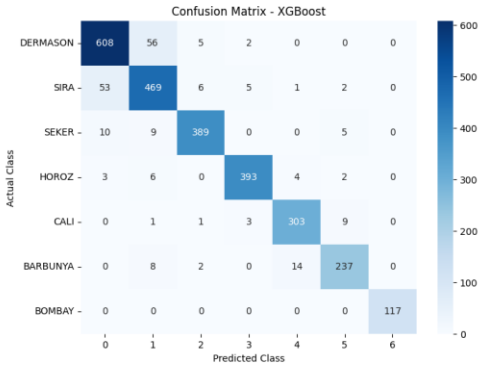


- Confusion Matrix untuk masing – masing model.

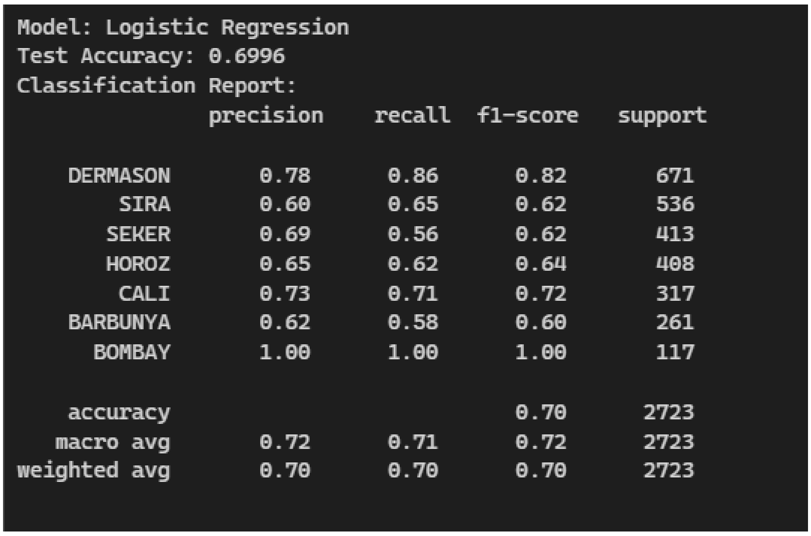

Confusion Matrix - Logistic Regression


Confusion Matrix - Decision Tree

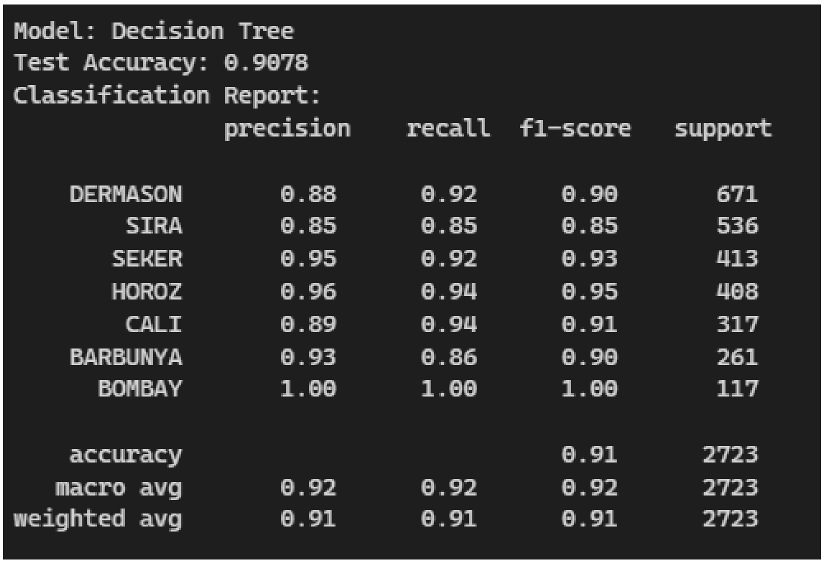

Confusion Matrix - K-Nearest Neighbors


Confusion Matrix - XGBoost

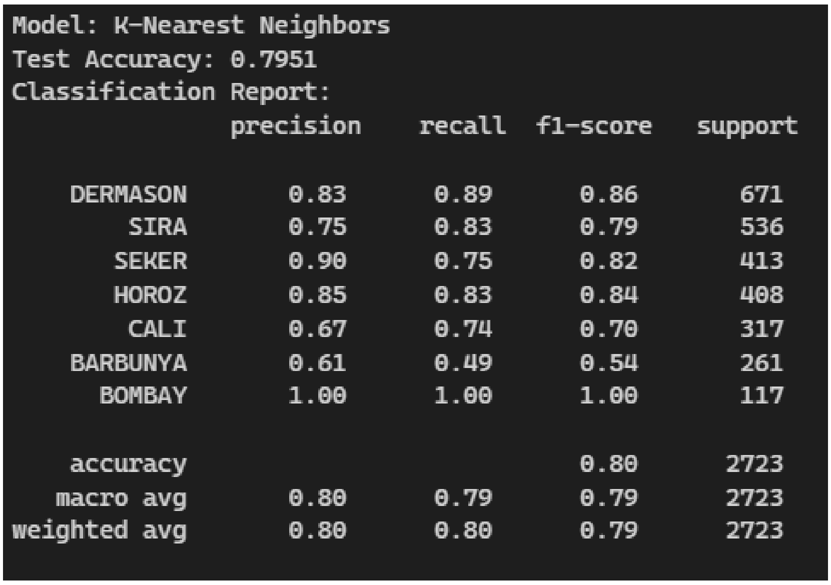
- Classification Report dari setiap model



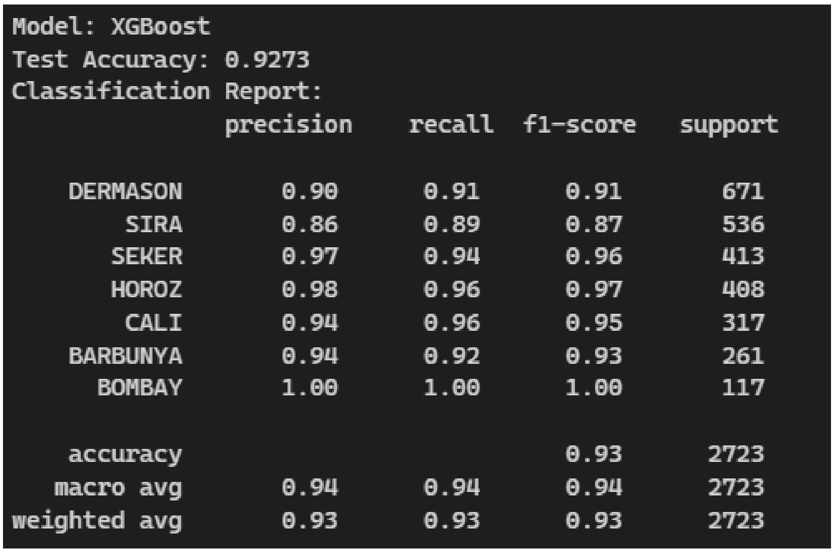
```
Model: Logistic Regression
Test Accuracy: 0.6996
Classification Report:
              precision    recall  f1-score   support

    DERMASON       0.78      0.86      0.82       671
        SIRA       0.60      0.65      0.62       536
       SEKER       0.69      0.56      0.62       413
       HOROZ       0.65      0.62      0.64       408
        CALI       0.73      0.71      0.72       317
    BARBUNYA       0.62      0.58      0.60       261
      BOMBAY       1.00      1.00      1.00       117

    accuracy                           0.70      2723
   macro avg       0.72      0.71      0.72      2723
weighted avg       0.70      0.70      0.70      2723
```



```
Model: Decision Tree
Test Accuracy: 0.9078
Classification Report:
              precision    recall  f1-score   support

    DERMASON       0.88      0.92      0.90       671
        SIRA       0.85      0.85      0.85       536
       SEKER       0.95      0.92      0.93       413
       HOROZ       0.96      0.94      0.95       408
        CALI       0.89      0.94      0.91       317
    BARBUNYA       0.93      0.86      0.90       261
      BOMBAY       1.00      1.00      1.00       117

    accuracy                           0.91      2723
   macro avg       0.92      0.92      0.92      2723
weighted avg       0.91      0.91      0.91      2723
```

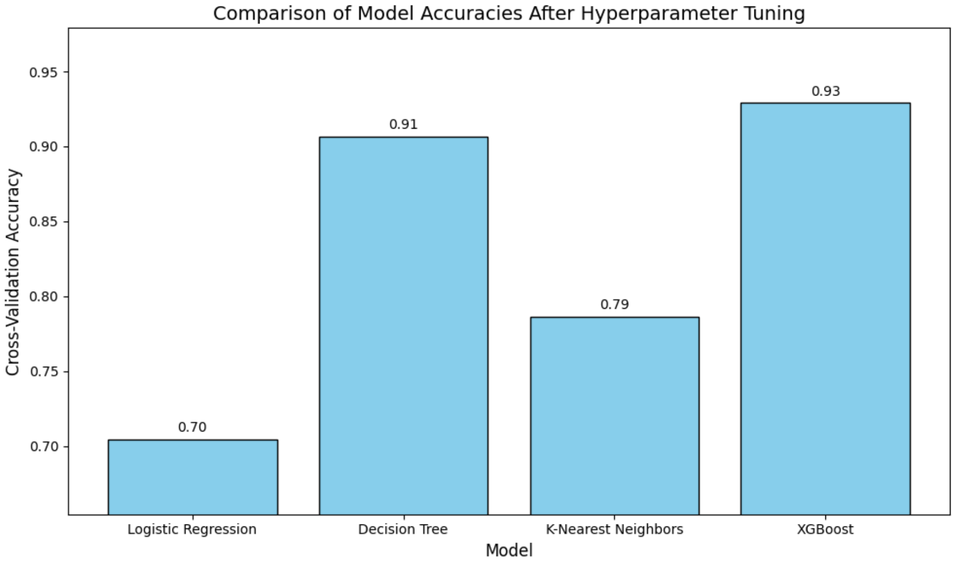


```
Model: K-Nearest Neighbors
Test Accuracy: 0.7951
Classification Report:
              precision    recall  f1-score   support

    DERMASON       0.83      0.89      0.86       671
        SIRA       0.75      0.83      0.79       536
       SEKER       0.90      0.75      0.82       413
       HOROZ       0.85      0.83      0.84       408
        CALI       0.67      0.74      0.70       317
    BARBUNYA       0.61      0.49      0.54       261
      BOMBAY       1.00      1.00      1.00       117

    accuracy                           0.80      2723
   macro avg       0.80      0.79      0.79      2723
weighted avg       0.80      0.80      0.79      2723
```

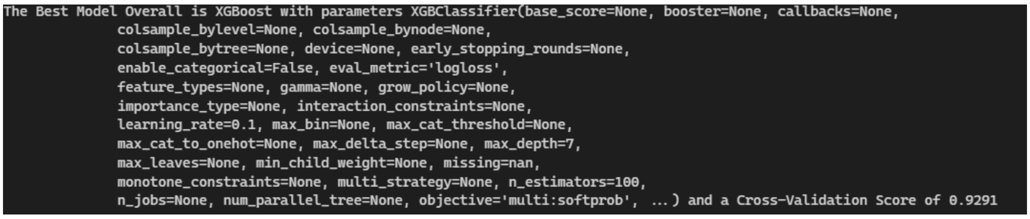


```
Model: XGBoost
Test Accuracy: 0.9273
Classification Report:
               precision    recall  f1-score   support

    DERMASON       0.90      0.91      0.91       671
        SIRA       0.86      0.89      0.87       536
       SEKER       0.97      0.94      0.96       413
       HOROZ       0.98      0.96      0.97       408
        CALI       0.94      0.96      0.95       317
    BARBUNYA       0.94      0.92      0.93       261
      BOMBAY       1.00      1.00      1.00       117

    accuracy                           0.93      2723
   macro avg       0.94      0.94      0.94      2723
weighted avg       0.93      0.93      0.93      2723
```

- Visualisasi hasil hypertuning



- Identifikasi model terbaik.



```
The Best Model Overall is XGBoost with parameters XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.1, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=7,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=100,
              n_jobs=None, num_parallel_tree=None, objective='multi:softprob',  ...) and a Cross-Validation Score of 0.9291
```

model XGBoost adalah model terbaik yang dapat digunakan pada dataset dengan parameter max_depth = 5, n_estimator = 200m dan learning rate = 0.01. Dengan nilai Cross Validation 0.9289 model ini menunjukkan performa yang sangat baik dalam memprediksi kelas pada parameter setelah dilakukan optimasi.

Link YouTube : https://youtu.be/G0XY6HPp7pI