

```
1 #%% md
2 ##1) Pengumpulan Data
3
4
5 Dataset yang digunakan adalah dataset yang bersumber
dari link berikut : https://archive.ics.uci.edu/
dataset/45/heart+disease
6 Dataset yang dipakai adalah dataset dengan nama file
"Hungarian.data" diharapkan sebelum memakai dataset
tersebut anda dapat membaca deskripsi dataset yang
ada di dalam file "heart-disease.names"
7
8 #%% md
9 ##2. Menelaah Data
10 Mengimpor modul-modul yang diperlukan, seperti pandas
, numpy, re, dan itertools.
11 Modul-modul ini digunakan untuk melakukan operasi
data, manipulasi string, dan iterasi.
12 #%%
13 import pandas as pd
14 import numpy as np
15 import re
16 import itertools
17 #%%
18 from google.colab import drive
19 drive.mount('/content/drive')
20 #%%
21 dir = '/content/drive/My Drive/BK/hungarian.data'
22 #%%
23 #dir adalah variabel yang digunakan untuk menyimpan
data
24 #dir = 'hungarian.data'
25 #%%
26 with open(dir, encoding='Latin1') as file:
27     lines = [line.strip() for line in file]
28
29 lines [0:10]
30 #%%
31 data = itertools.takewhile(
32     lambda x: len(x) == 76,
33     (''.join(lines[i:(i+10)]).split() for i in range
```

```
33 (0, len(lines), 10))
34 )
35
36 df = pd.DataFrame.from_records(data)
37 #%%
38 df.tail()
39 #%%
40 df.dtypes
41 #%%
42 df = df.iloc[:, :-1]
43 df = df.drop(df.columns[0], axis=1)
44 #%%
45 df = df.astype(float)
46 #%%
47 df.info()
48 #%% md
49 ##3) Validasi Data
50 Pada tahap ini bertujuan untuk mengetahui dan
memahami isi dari dataset agar dapat dilakukan
penanganan sesuai dengan kondisinya
51
52 mengubah nilai -9.0 menjadi nilai null value sesuai
dengan deskripsi dataset
53 #%%
54 df.replace(-9.0, np.nan, inplace=True)
55 #%% md
56 menghitung jumlah nilai null value
57 #%%
58 df.isnull().sum()
59 #%%
60 df.head()
61 #%%
62 df.info()
63 #%% md
64 ##4. Menentukan Object Data
65 Fitur adalah variabel yang digunakan untuk melakukan
analisis atau pemodelan data. Dari 74 kolom yang ada
, hanya dipilih 14 kolom yang dianggap relevan dan
memiliki cukup data yang tidak kosong.
66 #%%
67 df_selected = df.iloc[:, [1, 2, 7, 8, 10, 14, 17, 30, 36, 38,
```

```
67 39,42,49,56]]
68 #%%
69 df_selected.head()
70 #%%
71 df_selected.info()
72 #%%
73 column_mapping = {
74     2: 'age',
75     3: 'sex',
76     8: 'cp',
77     9: 'trestbps',
78     11: 'chol',
79     15: 'fbs',
80     18: 'restecg',
81     31: 'thalach',
82     37: 'exang',
83     39: 'oldpeak',
84     40: 'slope',
85     43: 'ca',
86     50: 'thal',
87     57: 'target',
88 }
89 #%%
90 df_selected.rename(columns=column_mapping, inplace=True)
91 #%%
92 df_selected.info()
93 #%%
94 df_selected.value_counts()
95 #%% md
96 ##5. Membersihkan Data
97 #%% md
98 Sebelum melakukan pemodelan, dilakukan pembersihan
    data agar model yang dihasilkan lebih akurat
99 #%% md
100 Koding dibawah ini digunakan untuk menghitung
    jumlah null values yang ada didalam dataset
101 #%%
102 df_selected.isnull().sum()
103 #%% md
104 karena ada data yang hampir 90% nya kosong (slope,
```

```
104 ca, thal) maka lebih baik dihapus menggunakan "drop"
105 #%%
106 columns_to_drop = ['ca','slope','thal']
107 df_selected = df_selected.drop(columns_to_drop, axis
=1)
108 #%%
109 df_selected.isnull().sum()
110 #%% md
111 Dikarenakan masih ada nilai null di beberapa kolom
fitur (trestbps, chol, fbs, restecg, thalach, exang
, oldpeak) maka akan dilakukan pengisian nilai null
menggunakan mean di setiap kolomnya.
112 #%%
113 meanTBPS = df_selected['trestbps'].dropna()
114 meanChol = df_selected['chol'].dropna()
115 meanfbs = df_selected['fbs'].dropna()
116 meanRestCG = df_selected['restecg'].dropna()
117 meanthalach = df_selected['thalach'].dropna()
118 meanexang = df_selected['exang'].dropna()
119
120 meanTBPS = meanTBPS.astype(float)
121 meanChol = meanChol.astype(float)
122 meanfbs = meanfbs.astype(float)
123 meanthalach = meanthalach.astype(float)
124 meanexang = meanexang.astype(float)
125 meanRestCG = meanRestCG.astype(float)
126
127 meanTBPS = round(meanTBPS.mean())
128 meanChol = round(meanChol.mean())
129 meanfbs = round(meanfbs.mean())
130 meanthalach = round(meanthalach.mean())
131 meanexang = round(meanexang.mean())
132 meanRestCG = round(meanRestCG.mean())
133
134 #%% md
135 Mengubah nilai null menjadi nilai mean yang sudah
ditentukan sebelumnya
136 #%%
137 fill_value = {'trestbps': meanTBPS, 'chol': meanChol
, 'fbs': meanfbs,
138           'thalach': meanthalach, 'exang':
```

```
138 meanexang, 'restecg':meanRestCG}
139 dfClean = df_selected.fillna(value=fill_value)
140 #%%
141 dfClean.info()
142 #%%
143 dfClean.isnull().sum()
144 #%% md
145 Pengecekan duplikasi data
146 #%%
147 duplicate_rows = dfClean.duplicated()
148 dfClean[duplicate_rows]
149 #%%
150 print("All Duplicate Rows:")
151 dfClean[dfClean.duplicated(keep=False)]
152 #%%
153 dfClean = dfClean.drop_duplicates()
154 print("All Duplicate Rows:")
155 dfClean[dfClean.duplicated(keep=False)]
156 #%%
157 dfClean.head()
158 #%%
159 dfClean['target'].value_counts()
160 #%%
161 import seaborn as sns
162 import matplotlib.pyplot as plt
163 #%% md
164 Mencari korelasi antar fitur
165 #%%
166 dfClean.corr()
167 #%%
168 cor_mat=dfClean.corr()
169 fig,ax=plt.subplots(figsize=(15,10))
170 sns.heatmap(cor_mat,annot=True,linewidths=0.5,fmt=".3f")
171 #%% md
172 ##6. Konstruksi Data
173 Dalam tahap ini, Konstruksi data bertujuan untuk  
menyesuaikan semua tipe data yang ada di dalam  
dataset. Namun pada tahap ini, dataset sudah  
memiliki tipe data yang sesuai sehingga tidak perlu  
dilakukan penyesuaian kembali
```

```
174 #%%
175 dfClean.info()
176 #%%
177 dfClean.head(5)
178 #%% md
179 Setelah Menyesuaikan tipe dataset, kita harus
    memisahkan antara fitur dan target lalu simpan
    kedalam variabel.
180 #%%
181 x = dfClean.drop("target",axis=1).values
182 y = dfClean.iloc[:, -1]
183 #%% md
184 Setelah memisahkan antara fitur dan target,
    sebaiknya kita melakukan pengecekan terlebih dahulu
    terhadap persebaran jumlah target terlebih dahulu
185 #%%
186 dfClean['target'].value_counts().plot(kind='bar',
    figsize=(10,6),color=['green','blue'])
187 plt.title("Count of the target")
188 plt.xticks(rotation=0);
189 #%% md
190 Pada grafik diatas, ditunjukkan bahwa persebaran
    jumlah target tidak seimbang, oleh karena itu perlu
    diseimbangkan terlebih dahulu.
191
192 Menyeimbangkan target ada 2 cara, yaitu oversampling
    dan undersampling. Oversampling dilakukan jika
    jumlah dataset sedikit, sedangkan undersampling
    dilakukan jika jumlah data terlalu banyak.
193
194 Disini kita akan melakukan oversampling karena
    jumlah data yang dimiliki tidak banyak.
195
196 Salah satu metode oversampling yang akan digunakan
    adalah SMOTE
197 #%%
198 from imblearn.over_sampling import SMOTE
199 #%%
200 #Oversampling
201 smote = SMOTE(random_state=42)
202 X_smote_resampled, y_smote_resampled = smote.
```

```
202 fit_resample(x, y)
203 #%%
204 plt.figure(figsize=(12, 4))
205
206 new_df1 = pd.DataFrame(data=y)
207
208 plt.subplot(1, 2, 1)
209 new_df1.value_counts().plot(kind='bar', figsize=(10, 6),
   ),color=['green','blue','red','yellow'])
210 plt.title("target before over sampling with SMOTE")
211 plt.xticks(rotation=0);
212
213 plt.subplot(1, 2, 2)
214 new_df2 = pd.DataFrame(data=y_smote_resampled)
215
216 new_df2.value_counts().plot(kind='bar', figsize=(10, 6),
   ),color=['green','blue','red','yellow'])
217 plt.title("target after over sampling with SMOTE")
218 plt.xticks(rotation=0);
219
220 plt.tight_layout()
221 plt.show()
222 #%% md
223 Pada Grafik diatas dapat dilihat ketika target belum
      di seimbangkan dan sudah diseimbangkan menggunakan
      oversampling
224 #%%
225 new_df1 = pd.DataFrame(data=y)
226 new_df1.value_counts()
227 #%%
228 #Oversampling
229 new_df1 = pd.DataFrame(data=y_smote_resampled)
230 new_df2.value_counts()
231 #%% md
232 Setelah menyeimbangkan persebaran jumlah target,
      kita akan melakukan pengecekan, apakah perlu
      dilakukan normalisasi/standarisasi pada dataset kita
      .
233 #%%
234 dfClean.describe()
235 #%% md
```

```
236 Pada deskripsi diatas, bisa dilihat bahwa terdapat  
rentang nilai yang cukup jauh pada standar deviasi  
setiap fitur dataset yang kita miliki, oleh karena  
itu perlu dilakukan normalisasi/standarisasi agar  
memperkecil rentang antara sandar deviasi setiap  
kolom.  
237 #%%  
238 from sklearn.preprocessing import MinMaxScaler  
239  
240 scaler = MinMaxScaler()  
241  
242 X_smote_resampled_normal = scaler.fit_transform(  
    X_smote_resampled)  
243  
244 len(X_smote_resampled_normal)  
245 #%%  
246 dfcek1 = pd.DataFrame(X_smote_resampled_normal)  
247 dfcek1.describe()  
248 #%% md  
249 ##SANGAT PENTING!!!  
250  
251 Setelah dilakukan normalisasi pada fitur,  
selanjutnya kita perlu membagi fitur dan target  
menjadi data train dan test.  
252 #%%  
253 from sklearn.model_selection import train_test_split  
254 #%%  
255 #membagi fitur dan target menjadi data train dan  
#test (untuk yang Oversample saja)  
256 X_train, X_test, y_train, y_test = train_test_split(  
    X_smote_resampled, y_smote_resampled, test_size=0.2  
    , random_state=42, stratify=y_smote_resampled)  
257 #%%  
258 #Membagi fitur dan target menjadi data train dan  
#test (untuk yang oversample + normalization)  
259 X_train_normal, X_test_normal, y_train_normal,  
y_test_normal = train_test_split(  
    X_smote_resampled_normal, y_smote_resampled,  
    test_size=0.2, random_state=42, stratify =  
    y_smote_resampled)  
260 #%% md
```

```
261 ## 7. Modelling
262 Pada tahap ini kita akan memulai untuk membangun
   sebuah model.
263
264
265 Dibawah ini merupakan sebuah fungsi untuk
   menampilkan hasil akurasi dan rata - rata dari
   recall , f1 dan precision score setiap model. Fungsi
   ini nantinya akan dipanggil di setiap model.
   Membuat Fungsi ini bersifat opsional.
266
267 #%%
268 from sklearn.metrics import accuracy_score,
   recall_score,f1_score,precision_score,roc_auc_score,
   confusion_matrix,precision_score
269 def evaluation(Y_test,Y_pred):
270     acc = accuracy_score(Y_test,Y_pred)
271     rcl = recall_score(Y_test,Y_pred,average = '
   weighted')
272     f1 = f1_score(Y_test,Y_pred,average = 'weighted')
273     ps = precision_score(Y_test,Y_pred,average = '
   weighted')
274
275     metric_dict={ 'accuracy': round(acc,3),
276                   'recall': round(rcl,3),
277                   'F1 score': round(f1,3),
278                   'Precision score': round(ps,3)
279                 }
280     return print(metric_dict)
281
282 #%% md
283 Pada tahap ini kita akan memulai untuk membangun
   sebuah model.
284
285
286 Dibawah ini merupakan sebuah fungsi untuk
   menampilkan hasil akurasi dan rata - rata dari
   recall , f1 dan precision score setiap model. Fungsi
   ini nantinya akan dipanggil di setiap model.
   Membuat Fungsi ini bersifat opsional.
287
```

```
288 #%%
289 from sklearn.neighbors import KNeighborsClassifier
290 from sklearn.ensemble import RandomForestClassifier
291 from xgboost import XGBClassifier
292 from sklearn.metrics import accuracy_score,
293     classification_report
294 knn_model = KNeighborsClassifier(n_neighbors = 3)
295 knn_model.fit(X_train, y_train)
296 #%% md
297 Berikut adalah kode program untuk menampilkan hasil
298 akurasi dengan algoritma KNN
299 #%%
300 y_pred_knn = knn_model.predict(X_test)
301 # Evaluate the KNN model
302 print("K-Nearest Neighbors (KNN) Model:")
303 accuracy_knn_smote = round(accuracy_score(y_test,
304     y_pred_knn), 3)
305 print("Accuracy:", accuracy_knn_smote)
306 print("Classification Report:")
307 print(classification_report(y_test, y_pred_knn))
308 #%% md
309 evaluation(y_test,y_pred_knn)
310 Pada visualisasi ini ditampilkan visualisasi
311 confusion matrix untuk membandingkan hasil prediksi
312 model dengan nilai sebenarnya.
313 #%%
314 cm = confusion_matrix(y_test, y_pred_knn)
315 plt.figure(figsize=(8, 6))
316 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
317 plt.title('Confusion Matrix')
318 plt.xlabel('True')
319 plt.ylabel('Predict')
320 plt.show()
321 #%% md
322 Selanjutnya kita akan membangun model dengan
323 algoritma random forest dengan n_estimators yaitu
324 100, n_estimators sendiri berguna
325 mengatur jumlah pohon keputusan yang akan dibangun
```

```
322
323 #%%
324 rf_model = RandomForestClassifier(n_estimators=100,
325 random_state=42)
325 rf_model.fit(X_train, y_train)
326 #%%
327 y_pred_rf = rf_model.predict(X_test)
328
329 # Evaluate the Random Forest model
330 print("\nRandom Forest Model:")
331 accuracy_rf_smote = round(accuracy_score(y_test,
332 y_pred_rf), 3)
332 print("Accuracy:", accuracy_rf_smote)
333 print("Classification Report:")
334 print(classification_report(y_test, y_pred_rf))
335
336 #%%
337 evaluation(y_test, y_pred_rf)
338 #%%
339 cm = confusion_matrix(y_test, y_pred_rf)
340
341 plt.figure(figsize=(8, 6))
342 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
343 plt.title('Confusion Matrix')
344 plt.xlabel('True')
345 plt.ylabel('Predict')
346 plt.show()
347
348 #%% md
349 ##XGBoost
350
351
352 Pada tahap ini dalam membangun model, kita akan menggunakan algoritma XGBoost dengan learning rate yaitu 0.1. learning rate berguna untuk mengontrol seberapa besar kita menyesuaikan bobot model.
353
354 #%%
355 xgb_model = XGBClassifier(learning_rate=0.1,
356 n_estimators=100, random_state=42)
356 xgb_model.fit(X_train, y_train)
```

```
357 #%%
358 y_pred_xgb = xgb_model.predict(X_test)
359
360 # Evaluate the XGBoost model
361 print("\nXGBoost Model:")
362 accuracy_xgb_smote = round(accuracy_score(y_test,
363 y_pred_xgb), 3)
363 print("Accuracy:", accuracy_xgb_smote)
364 print("Classification Report:")
365 print(classification_report(y_test, y_pred_xgb))
366
367 #%%
368 evaluation(y_test, y_pred_xgb)
369 #%%
370 cm = confusion_matrix(y_test, y_pred_xgb)
371
372 plt.figure(figsize=(8, 6))
373 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
374 plt.title('Confusion Matrix')
375 plt.xlabel('True')
376 plt.ylabel('Predict')
377 plt.show()
378
379 #%% md
380 ##Oversample + Normalisasi
381 Pada bagian ini kita akan membuat sebuah model yang dimana data yang dipakai kali ini yang sudah dilakukan oversample dan normalisasi. Algoritma yang digunakan sama seperti sebelumnya yaitu KNN, Random Forest, dan XGBoost. Sekaligus dibuat visualisasi hasil evaluasi pada masing-masing model.
382
383 ##KNN
384 #%%
385 from sklearn.neighbors import KNeighborsClassifier
386 from sklearn.ensemble import RandomForestClassifier
387 from xgboost import XGBClassifier
388 from sklearn.metrics import accuracy_score,
389 classification_report
390
```

```
391 knn_model = KNeighborsClassifier(n_neighbors=3)
392 knn_model.fit(X_train_normal, y_train_normal)
393
394 #%%
395 y_pred_knn = knn_model.predict(X_test_normal)
396
397 # Evaluate the KNN model
398 print("K-Nearest Neighbors (KNN) Model:")
399 accuracy_knn_smote_normal = round(accuracy_score(
    y_test_normal,y_pred_knn),3)
400 print("Accuracy:", accuracy_knn_smote_normal)
401 print("Classification Report:")
402 print(classification_report(y_test_normal,
    y_pred_knn))
403
404 #%%
405 evaluation(y_test_normal,y_pred_knn)
406 #%%
407 cm = confusion_matrix(y_test_normal, y_pred_knn)
408 plt.figure(figsize=(8, 6))
409 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
410 plt.title('Confusion Matrix')
411 plt.xlabel('True')
412 plt.ylabel('Predict')
413 plt.show()
414
415 #%% md
416 ##Random Forest
417 #%%
418 y_pred_rf = rf_model.predict(X_test_normal)
419
420 # Evaluate the Random Forest model
421 print("\nRandom Forest Model:")
422 accuracy_rf_smote_normal = round(accuracy_score(
    y_test_normal, y_pred_rf),3)
423 print("Accuracy:",accuracy_rf_smote_normal )
424 print("Classification Report:")
425 print(classification_report(y_test_normal, y_pred_rf
    ))
426
427 #%%
```

```
428 evaluation(y_test_normal,y_pred_rf)
429 #%%
430 cm = confusion_matrix(y_test_normal, y_pred_rf)
431 plt.figure(figsize=(8, 6))
432 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
433 plt.title('Confusion Matrix')
434 plt.xlabel('True')
435 plt.ylabel('Predict')
436 plt.show()
437
438 #%% md
439 ##XGBoost
440 #%%
441 xgb_model = XGBClassifier(learning_rate=0.1,
    n_estimators=100, random_state=42)
442 xgb_model.fit(X_train_normal, y_train_normal)
443 #%%
444 y_pred_xgb = xgb_model.predict(X_test_normal)
445 # Evaluate the XGBoost model
446 print("\nXGBoost Model:")
447 accuracy_xgb_smote_normal = round(accuracy_score(
    y_test_normal, y_pred_xgb),3)
448 print("Accuracy:",accuracy_xgb_smote_normal)
449 print("Classification Report:")
450 print(classification_report(y_test_normal,
    y_pred_xgb))
451
452 #%%
453 evaluation(y_test_normal,y_pred_xgb)
454 #%%
455 cm = confusion_matrix(y_test_normal, y_pred_xgb)
456
457 plt.figure(figsize=(8, 6))
458 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
459 plt.title('Confusion Matrix')
460 plt.xlabel('True')
461 plt.ylabel('Predict')
462 plt.show()
463
464 #%% md
465 ##Tunning + Normalization + Oversample
```

```
466 Pada pembuatan model kali ini masih menggunakan  
algoritma yang sama (KNN, Random Forest, dan XGBoost  
, namun data yang digunakan adalah data yang sudah  
dilakukan TunNING Parameter, Normalisasi, dan  
Oversample.  
467  
468 ##KNN  
469 #%%  
470  
471 from sklearn.neighbors import KNeighborsClassifier  
472 from sklearn.ensemble import RandomForestClassifier  
473 from xgboost import XGBClassifier  
474 from sklearn.metrics import accuracy_score,  
classification_report  
475 from sklearn.model_selection import  
RandomizedSearchCV  
476  
477 #%% md  
478 Setiap parameter tunnning tidak selalu sama karena  
bergantung pada algoritma yang digunakan.  
479 #%%  
480 knn_model = KNeighborsClassifier()  
481  
482 param_grid = {  
483     "n_neighbors": range(3, 21),  
484     "metric": ["euclidean", "manhattan", "chebyshev"]  
},  
485     "weights": ["uniform", "distance"],  
486     "algorithm": ["auto", "ball_tree", "kd_tree"],  
487     "leaf_size": range(10, 61),  
488 }  
489  
490 knn_model = RandomizedSearchCV(estimator=knn_model,  
param_distributions=param_grid, n_iter=100, scoring=  
"accuracy", cv=5)  
491 #%%  
492 knn_model.fit(X_train_normal, y_train_normal)  
493  
494 best_params = knn_model.best_params_  
495 print(f"Best parameters: {best_params}")  
496
```

```
497 #%%
498 y_pred_knn = knn_model.predict(X_test_normal)
499 #%%
500 # Evaluate the KNN model
501 print("K-Nearest Neighbors (KNN) Model:")
502 accuracy_knn_smote_normal_Tun = round(accuracy_score
(y_test_normal,y_pred_knn),3)
503 print("Accuracy:", accuracy_knn_smote_normal_Tun)
504 print("Classification Report:")
505 print(classification_report(y_test_normal,
y_pred_knn))
506 #%%
507 evaluation(y_test_normal,y_pred_knn)
508 #%%
509 cm = confusion_matrix(y_test_normal, y_pred_knn)
510
511 plt.figure(figsize=(8, 6))
512 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
513 plt.title('Confusion Matrix')
514 plt.xlabel('True')
515 plt.ylabel('Predict')
516 plt.show()
517
518 #%% md
519 ##RandomForest
520 #%%
521 rf_model = RandomForestClassifier()
522
523 param_grid = {
524     "n_estimators": [100, 200],
525     "max_depth": [10, 15],
526     "min_samples_leaf": [1, 2],
527     "min_samples_split": [2, 5],
528     "max_features": ["sqrt", "log2"],
529     # "random_state": [42, 100, 200]
530 }
531
532 rf_model = RandomizedSearchCV(rf_model, param_grid,
n_iter=100, cv=5, n_jobs=-1)
533 #%%
534 rf_model.fit(X_train_normal, y_train_normal)
```

```
535
536 best_params = rf_model.best_params_
537 print(f"Best parameters: {best_params}")
538
539 #%%
540 y_pred_rf = rf_model.predict(X_test_normal)
541
542 # Evaluate the Random Forest model
543 print("\nRandom Forest Model:")
544 accuracy_rf_smote_normal_Tun = round(accuracy_score(
    y_test_normal, y_pred_rf), 3)
545 print("Accuracy:", accuracy_rf_smote_normal_Tun)
546 print("Classification Report:")
547 print(classification_report(y_test_normal, y_pred_rf))
548
549 #%%
550 evaluation(y_test_normal, y_pred_rf)
551 #%%
552 cm = confusion_matrix(y_test_normal, y_pred_knn)
553
554 plt.figure(figsize=(8, 6))
555 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
556 plt.title('Confusion Matrix')
557 plt.xlabel('True')
558 plt.ylabel('Predict')
559 plt.show()
560 #%% md
561 ##XGBoost
562 #%%
563 xgb_model = XGBClassifier()
564
565 param_grid = {
566     "max_depth": [3, 5, 7],
567     "learning_rate": [0.01, 0.1],
568     "n_estimators": [100, 200],
569     "gamma": [0, 0.1],
570     "colsample_bytree": [0.7, 0.8],
571 }
572
573 xgb_model = RandomizedSearchCV(xgb_model, param_grid
```

```
573 , n_iter=10, cv=5, n_jobs=-1)
574 #%%
575 xgb_model.fit(X_train_normal, y_train_normal)
576
577 best_params = xgb_model.best_params_
578 print(f"Best parameters: {best_params}")
579 #%%
580 y_pred_xgb = xgb_model.predict(X_test_normal)
581
582 # Evaluate the XGBoost model
583 print("\nXGBoost Model:")
584 accuracy_xgb_smote_normal_Tun = round(accuracy_score(
585     y_test_normal, y_pred_xgb), 3)
586 print("Accuracy:", accuracy_xgb_smote_normal_Tun)
587 print("Classification Report:")
588 print(classification_report(y_test_normal,
589     y_pred_xgb))
590
591 #%%
592 cm = confusion_matrix(y_test_normal, y_pred_xgb)
593
594 plt.figure(figsize=(8, 6))
595 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
596 plt.title('Confusion Matrix')
597 plt.xlabel('True')
598 plt.ylabel('Predict')
599 plt.show()
600
601 #%% md
602 ##8. Evaluasi
603 Selanjutnya kita akan melakukan evaluasi data
       sekaligus membandingkan antar algoritma guna dengan
       tujuan mengetahui jenis model algoritma yang
       menghasilkan hasil akurasi terbaik.
604 #%%
605 import matplotlib.pyplot as plt
606
607
608 model_comp1 = pd.DataFrame({'Model': ['K-Nearest
```

```
608 Neighbour', 'Random Forest',
609                                         'XGBoost'], '
610
611 model_comp1.head()
612 #%%
613 # Membuat bar plot dengan keterangan jumlah fig, ax
614 bars = plt.bar(model_comp1['Model'], model_comp1[
615     'Accuracy'], color=['red', 'green', 'blue'])
616 plt.xlabel('Model')
617 plt.ylabel('Accuracy (%)')
618 plt.title('Oversample')
619 plt.xticks(rotation=45, ha='right') # Untuk memutar
620 # label sumbu x agar lebih mudah dibaca
621 # Menambahkan keterangan jumlah di atas setiap bar
622 for bar in bars:
623     yval = bar.get_height()
624     plt.text(bar.get_x() + bar.get_width()/2, yval,
625             round(yval, 2), ha='center', va='bottom')
626 plt.show()
627 #%%
628 model_comp2 = pd.DataFrame({'Model': ['K-Nearest
629 Neighbour', 'Random Forest',
630                                         'XGBoost'],
631                                         'Accuracy': [accuracy_knn_smote_normal*100,
632
633 accuracy_rf_smote_normal*100,
634 accuracy_xgb_smote_normal*100]})
```

```

638 plt.xticks(rotation=45, ha='right') # Untuk memutar
    label sumbu x agar lebih mudah dibaca
639
640
641 # Menambahkan keterangan jumlah di atas setiap bar
642 for bar in bars:
643     yval = bar.get_height()
644     plt.text(bar.get_x() + bar.get_width()/2, yval,
645               round(yval, 2), ha='center', va='bottom')
646 plt.show()
647 #%%
648 model_comp3 = pd.DataFrame({'Model': ['K-Nearest
    Neighbour', 'Random Forest',
                                             'XGBoost'],
649                               'Accuracy': [accuracy_knn_smote_normal_Tun*100,
650                                            accuracy_rf_smote_normal_Tun*100,
651                                            accuracy_xgb_smote_normal_Tun*100]})

651 model_comp3.head()
652 #%%
653 # Data frame
654 model_compBest = pd.DataFrame({
655     'Model': ['K-Nearest Neighbour OverSample
        Tunning', 'Random Forest OverSample',
656                 'XGB OverSample Standarization Tunning
        '],
657     'Accuracy': [accuracy_knn_smote_normal_Tun*100,
658                   accuracy_rf_smote_normal*100,
659                   accuracy_xgb_smote_normal_Tun*100]
660 })
661 # Membuat bar plot dengan keterangan jumlah
662 fig, ax = plt.subplots()
663 bars = plt.bar(model_compBest['Model'],
664                  model_compBest['Accuracy'], color=['red', 'green', 'blue'])
665 plt.xlabel('Model')
666 plt.ylabel('Accuracy (%)')
667 plt.title('Best Model Comparison')
668 plt.xticks(rotation=45, ha='right') # Untuk memutar

```

```
666 label sumbu x agar lebih mudah dibaca
667
668 # Menambahkan keterangan jumlah di atas setiap bar
669 for bar in bars:
670     yval = bar.get_height()
671     plt.text(bar.get_x() + bar.get_width()/2, yval,
672               round(yval, 2), ha='center', va='bottom')
673
674 plt.show()
675 %% md
676 ##9. Streamlit
677 %% md
678 ##10. Kesimpulan
679
680
681
682
```