PROJECT-BASED ASSIGNMENT

PEMBELAJARAN MESIN

Oleh :

Johanes Raphael Nandaputra - 1301204243

Muhamad Fachri Haikal - 1301202398

Muhammad Naufal Abdillah - 1301201586

Rasyid Riyaldi - 1301200457

Kelompok 10

IF-44-01

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

2023

DAFTAR ISI

BAB I

FORMULASI MASALAH

Kelompok kami mendapat permasalahan Ensemble Learning dengan metode Bagging untuk mengklasifikasikan kelas/kategori dari calon kreditur (apakah akan menjadi kreditur yang baik atau buruk) berdasarkan profil calon kreditur yang diberikan yang diwakili oleh atribut-atribut seperti status pekerjaan, status perkawinan, tujuan kredit, usia, jenis kelamin, dll yang didapat dari dataset German credit.

BAB II

EKSPLORASI DAN PRA-PEMROSESAN DATA

Eksplorasi yang kami lakukan pada dataset German credit memberitahukan bahwa dataset ini memiliki 1000 baris dan 21 kolom, yang dimana hampir semua atribut merupakan data kategori / kualitatif, sehingga data-data tersebut perlu kita ubah terlebih dahulu menjadi data kuantitatif agar pemrosesan dapat berjalan dengan lancar. Maka dari itu, kami melakukan beberapa pra-pemrosesan data yang dianggap penting untuk dilakukan, yaitu:

1. Missing Value

Pada mengecek missing value di dataset German credit, kita memperoleh hasil yang menyatakan bahwa tidak terdapat missing value di tiap kolom. Jadi, kita tidak perlu melakukan penanganan pada *missing value*.

```
status                    False
duration                  False
credit_history            False
purpose                   False
amount                    False
savings                   False
employment_duration       False
installment_rate          False
personal_status_sex       False
other_debtors             False
present_residence         False
property                  False
age                       False
other_installment_plans   False
housing                   False
number_credits            False
job                       False
people_liable             False
telephone                 False
foreign_worker            False
credit_risk               False
dtype: bool
```

2. Ubah Value Kategori Menjadi Value Numerik

Seperti yang dinyatakan sebelumnya, agar pemrosesan dapat berjalan dengan lancar, maka akan diubah tipe data yang tadinya kategori / kualitatif menjadi numerik / kuantitatif. Langkah yang dilakukan adalah sebagai berikut:

a. Mengecek tipe data tiap kolom

Untuk mengetahui kolom mana saja dengan tipe data kategori, kita dapat melihatnya melalui tabel dengan kolom' Dtype' berikut.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   status                   1000 non-null   object
 1   duration                 1000 non-null   int64
 2   credit_history           1000 non-null   object
 3   purpose                  1000 non-null   object
 4   amount                   1000 non-null   int64
 5   savings                  1000 non-null   object
 6   employment_duration      1000 non-null   object
 7   installment_rate         1000 non-null   object
 8   personal_status_sex      1000 non-null   object
 9   other_debtors            1000 non-null   object
 10  present_residence        1000 non-null   object
 11  property                 1000 non-null   object
 12  age                      1000 non-null   int64
 13  other_installment_plans  1000 non-null   object
 14  housing                  1000 non-null   object
 15  number_credits           1000 non-null   object
 16  job                      1000 non-null   object
 17  people_liable            1000 non-null   object
 18  telephone                1000 non-null   object
 19  foreign_worker           1000 non-null   object
 20  credit_risk              1000 non-null   object
dtypes: int64(3), object(18)
memory usage: 164.2+ KB
```

b. Mencari semua kolom dengan value kategori

Setelah dilakukan pengecekan, kolom tersebut akan disimpan ke list *num_cols* untuk membantu langkah selanjutnya. Berikut ditampilkan 18 kolom mana saja yang memiliki tipe data kategori.

```
['number_credits',
 'other_debtors',
 'credit_history',
 'property',
 'purpose',
 'other_installment_plans',
 'savings',
 'people_liable',
 'personal_status_sex',
 'employment_duration',
 'status',
 'installment_rate',
 'housing',
 'telephone',
 'foreign_worker',
 'credit_risk',
 'job',
 'present_residence']
```

c. Mencari value unik pada kolom dengan value kategori

Untuk mengetahui kategori dengan value apa saja yang harus kita ubah menjadi numerik, kita perlu mencari value yang unik dari tiap kolom untuk memudahkan penggantiannya. Berikut didapatkan value apa saja terdapat di tiap kolom dengan tipe data kategori.

```
number_credits : ['1', '03-Feb', '05-Apr', '>= 6'] , (4)
other_debtors : ['none', 'guarantor', 'co-applicant'] , (3)
credit_history : ['all credits at this bank paid back duly', 'no credits taken/all credits paid back duly', 'existing credits pai
property : ['car or other', 'unknown / no property', 'building soc. savings agr./life insurance', 'real estate'] , (4)
purpose : ['car (used)', 'others', 'retraining', 'furniture/equipment', 'car (new)', 'business', 'domestic appliances', 'radio/te
other_installment_plans : ['none', 'bank', 'stores'] , (3)
savings : ['unknown/no savings account', '... <  100 DM', '100 <= ... <  500 DM', '... >= 1000 DM', '500 <= ... < 1000 DM'] , (5)
people_liable : ['0 to 2', '3 or more'] , (2)
personal_status_sex : ['female : non-single or male : single', 'male : married/widowed', 'female : single', 'male : divorced/sepa
employment_duration : ['< 1 yr', '1 <= ... < 4 yrs', '4 <= ... < 7 yrs', 'unemployed', '>= 7 yrs'] , (5)
status : ['no checking account', '... < 0 DM', '... >= 200 DM / salary for at least 1 year', '0<= ... < 200 DM'] , (4)
installment_rate : ['< 20', '25 <= ... < 35', '20 <= ... < 25', '>= 35'] , (4)
housing : ['for free', 'rent', 'own'] , (3)
telephone : ['no', 'yes (under customer name)'] , (2)
foreign_worker : ['no', 'yes'] , (2)
credit_risk : ['good', 'bad'] , (2)
job : ['skilled employee/official', 'unskilled - resident', 'unemployed/unskilled - non-resident', 'manager/self-empl./highly qua
present_residence : ['>= 7 yrs', '1 <= ... < 4 yrs', '4 <= ... < 7 yrs', '< 1 yr'] , (4)
```

d. Mengganti value tiap kolom dari kategori menjadi numerik

Setelah didapat value unik di tiap kolom kategori, maka selanjutnya akan dilakukan pergantian valuenya. Pergantian value akan dilakukan secara manual, dimana tiap value yang sudah diurutkan akan di *replace* dengan numerik 0 hingga banyaknya value unik tersebut. Misal, terdapat 3 value unik di suatu kolom, maka value unik pertama menjadi numerik 0, value unik kedua menjadi numerik 1, dan value unik terakhir menjadi numerik 2.

```
# Kolom 'present_residence'
df['present_residence'].replace(['>= 7 yrs', '1 <= ... < 4 yrs', '4 <= ... < 7 yrs', '< 1 yr'], [i for i in range(4)], inplace=

# Kolom 'credit_risk'
df['credit_risk'].replace(['good', 'bad'], [i for i in range(2)], inplace=True)

# Kolom 'people_liable'
df['people_liable'].replace(['0 to 2', '3 or more'], [i for i in range(2)], inplace=True)

# Kolom 'other_debtors'
df['other_debtors'].replace(['none', 'guarantor', 'co-applicant'], [i for i in range(3)], inplace=True)
```

```
# Kolom 'savings'
df['savings'].replace(['unknown/no savings account', '... <  100 DM', '100 <= ... <  500 DM', '... >= 1000 DM', '500 <= ... <

# Kolom 'foreign_worker'
df['foreign_worker'].replace(['no', 'yes'], [i for i in range(2)], inplace=True)

# Kolom 'employment_duration'
df['employment_duration'].replace(['< 1 yr', '1 <= ... < 4 yrs', '4 <= ... < 7 yrs', 'unemployed', '>= 7 yrs'], [i for i in ra

# Kolom 'property'
df['property'].replace(['car or other', 'unknown / no property', 'building soc. savings agr./life insurance', 'real estate'],
```

```
# Kolom 'telephone'
df['telephone'].replace(['no', 'yes (under customer name)'], [i for i in range(2)], inplace=True)

# Kolom 'housing'
df['housing'].replace(['for free', 'rent', 'own'], [i for i in range(3)], inplace=True)

# Kolom 'job'
df['job'].replace(['skilled employee/official', 'unskilled - resident', 'unemployed/unskilled - non-resident', 'manager/

# Kolom 'personal_status_sex'
df['personal_status_sex'].replace(['female : non-single or male : single', 'male : married/widowed', 'female : single',
```

```
# Kolom 'status'
df['status'].replace(['no checking account', '... < 0 DM', '... >= 200 DM / salary for at least 1 year', '0<= ... <

# Kolom 'number_credits'
df['number_credits'].replace(['1', '03-Feb', '05-Apr', '>= 6'], [i for i in range(4)], inplace=True)

# Kolom 'present_residence'
df['present_residence'].replace(['>= 7 yrs', '1 <= ... < 4 yrs', '4 <= ... < 7 yrs', '< 1 yr'], [i for i in range(4

# Kolom 'other_installment_plans'
df['other_installment_plans'].replace(['none', 'bank', 'stores'], [i for i in range(3)], inplace=True)
```

```
# Kolom 'credit_history'
df['credit_history'].replace(['all credits at this bank paid back duly', 'no credits taken/all credits paid bac

# Kolom 'installment_rate'
df['installment_rate'].replace(['< 20', '25 <= ... < 35', '20 <= ... < 25', '>= 35'], [i for i in range(4)], in

# Kolom 'purpose'
df['purpose'].replace( ['car (used)', 'others', 'retraining', 'furniture/equipment', 'car (new)', 'business',
```

e. Mengecek tipe data tiap kolom untuk melihat perubahan

Setelah dilakukan penggantian, maka hasilnya dapat dilihat seperti berikut, dari mayoritas tipe awalnya berupa kategorik menjadi numerik semua.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   status                  1000 non-null   int64
 1   duration                1000 non-null   int64
 2   credit_history          1000 non-null   int64
 3   purpose                 1000 non-null   int64
 4   amount                  1000 non-null   int64
 5   savings                 1000 non-null   int64
 6   employment_duration     1000 non-null   int64
 7   installment_rate        1000 non-null   int64
 8   personal_status_sex     1000 non-null   int64
 9   other_debtors           1000 non-null   int64
 10  present_residence       1000 non-null   int64
 11  property                1000 non-null   int64
 12  age                     1000 non-null   int64
 13  other_installment_plans 1000 non-null   int64
 14  housing                 1000 non-null   int64
 15  number_credits          1000 non-null   int64
 16  job                     1000 non-null   int64
 17  people_liable           1000 non-null   int64
 18  telephone               1000 non-null   int64
 19  foreign_worker          1000 non-null   int64
 20  credit_risk             1000 non-null   int64
dtypes: int64(21)
memory usage: 164.2 KB
```

3. Feature Transform (Scaling)

Jadi jika data dalam kondisi yang memiliki titik data yang berjauhan satu sama lain atau interval nilai yang berbeda cukup jauh, *scaling* adalah teknik untuk mendekatkan nilai data satu sama lain atau dengan kata lebih sederhana, *scaling*

digunakan untuk membuat titik data digeneralisasi sehingga jarak antara mereka akan lebih dekat. *Scaling* disini menggunakan standarisasi yang lebih tahan terhadap outlier dibandingkan metode scaling yang lainnya.

```
# Transformasi fitur menggunakan standardisasi
df_before_scaling = df.drop(columns=['credit_risk'])
scaler = StandardScaler()
df_after_scaling = pd.DataFrame(scaler.fit_transform(df_before_scaling), columns=df_before_scaling.columns)
```

Before Scaling:

	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex	other_debtors	present_residence	property	age
0	0	18	0	0	1049	0	0	0	0	0	0	0	21
1	0	9	0	1	2799	0	1	1	1	0	1	1	36
2	1	12	1	2	841	1	2	1	0	0	0	1	23
3	0	12	0	1	2122	0	1	2	1	0	1	1	39
4	0	12	0	1	2171	0	1	0	1	0	0	0	38

After Scaling:

	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex	other_debtors	present_residence	property	age
0	-1.344000	-0.240857	-1.031578	-1.153949	-0.787657	-0.717221	-1.304576	-0.879250	-1.150143	-0.301109	-0.957522	-1.398566	-1.281573
1	-1.344000	-0.987573	-1.031578	-0.680437	-0.167384	-0.717221	-0.612493	0.043363	0.153874	-0.301109	0.003845	-0.405266	0.040363
2	-0.265348	-0.738668	-0.022206	-0.206925	-0.861381	0.022182	0.079590	0.043363	-1.150143	-0.301109	-0.957522	-0.405266	-1.105315
3	-1.344000	-0.738668	-1.031578	-0.680437	-0.407341	-0.717221	-0.612493	0.965976	0.153874	-0.301109	0.003845	-0.405266	0.304750
4	-1.344000	-0.738668	-1.031578	-0.680437	-0.389974	-0.717221	-0.612493	-0.879250	0.153874	-0.301109	-0.957522	-1.398566	0.216621

4. Split Dataset

Sebelum masuk pada pemodelan, dataset akan dipisah dulu kedalam dua bagian, yaitu bagian untuk training, dan bagian untuk testing. Pembagian dari data train dan data test adalah 8:2, dan random state akan menggunakan NIM dari salah satu anggota kelompok.

```
# x: semua kolom kecuali kolom terakhir
x = df_after_scaling

# y: kolom terakhir (credit_risk)
y = df.iloc[:, -1]

# Ratio split 80 train / 20 test, seed: NIM salah satu anggota
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8,random_state=1301204243)
```

```
# Menampilkan hasil split data
print(f'Jumlah training data : {len(x_train)}')
print(f'Jumlah testing data  : {len(x_test)}')
```

```
Jumlah training data : 800
Jumlah testing data  : 200
```

BAB III

PEMODELAN

1. Parameter

Dalam model, ada beberapa parameter yang akan dimasukkan, sehingga pada saat eksplorasi, akan digunakkan beberapa parameter yang berbeda yang menyebabkan bermacam - macam hasil.

- Parameter *random_state* berguna untuk mengatur keacakan estimator. Pada parameter random_state, NIM - NIM anggota kelompok akan dijadikan sebagai seed dari modelnya.
- Parameter *criterion* berguna untuk mengatur fungsi yang mengontrol kualitas split atau percabangan pohon. Pada parameter criterion, akan dicoba dua, yaitu ‘entropy’ dan ‘gini’.
- Parameter *max_depth* berguna untuk mengatur tingkat kedalaman tree. Pada parameter max_depth, akan dicoba beberapa patokan kedalaman mulai dari kosong sampai 100.
- Parameter *max_features* berguna untuk mengatur maksimal fitur atau atribut yang digunakan pada tree. Pada parameter feature selection(max_feature), akan dicoba beberapa nilai, mulai dari 2 sampai ‘auto’.
- Parameter *n_estimators* berguna untuk mengatur jumlah tree pada bagging classifier. Pada parameter bagging n_estimators, akan dicoba nilai mulai dari 10 sampai 180.

```
# Parameter
seed = [1301204243, 1301202398, 1301201586, 1301200457]
criterion = ['entropy', 'gini']
max_depth = [20, 40, 60, 70, 80, 100, None]
max_features = [2,4,6,8,10,12,14, 'auto']
n_estimators = [10, 30, 50, 80, 100, 150, 180]
```

2. Model

```
# Decision Tree
dtree = DecisionTreeClassifier(criterion = criterion[0],
                               max_depth = max_depth[3],
                               max_features = max_features[1],
                               random_state = seed[0])

# Bagging classifier
model = BaggingClassifier(base_estimator = dtree,
                          n_estimators = n_estimators[2],
                          random_state = seed[0])
model.fit(x_train, y_train)
```

Pembuatan model klasifikasi *bagging* dimulai dengan membuat *Decision Tree* menggunakan parameter yang sudah didefinisikan sebelumnya. Kemudian *Decision Tree* tersebut dimasukkan ke dalam *Bagging Classifier* sehingga membentuk salah satu model klasifikasi *bagging*, yaitu *Random Forest*. Kemudian masukkan training data pada bagging model untuk membuat model trainingnya.

3. Hasil Pemodelan

Untuk model diatas, didapat bahwa akurasinya adalah sebesar 78.5%

```
# Akurasi model
accuracy_score(y_test, y_predict)
```

```
0.785
```

Akurasi model dapat dihitung menggunakan data yang ada pada confusion matrix berikut, ada sebanyak 138 yang berupa True Positive, 8 yang berupa False Positive, 35 yang berupa False Negative, dan 19 yang True Negative.

	Predicted Good	Predicted Bad
Good	138	8
Bad	35	19

Akurasi = (TP + TN) / (TP + TN + FP + FN) = (138 + 19) / (138 + 19 + 35 + 8) = 0.785

BAB IV

EVALUASI

Pada evaluasi, akan digunakan K-Fold cross validation untuk melihat score accuracy dari model. Sebelum itu, akan dicari n_splits yang paling ideal untuk digunakkan pada algoritma evaluasi dan juga pada saat eksperimentasi dan eksplorasi lanjut.

Untuk melihat rata - rata dari score, akan dicek cross validation score dari model, dengan menggunakan cross validation LeaveOneOut() yang setara dengan K Fold dengan n splits=n.

```
def evaluate_model(cv):
  scores = cross_val_score(model, x, y, scoring='accuracy',cv=cv,n_jobs=-1)
  return np.mean(scores), scores.min(), scores.max()
```

```
ideal, _, _ = evaluate_model(LeaveOneOut())
print('Ideal: %.3f' % ideal)
```

Kemudian, akan dilakukan looping dengan range 2 sampai 30 untuk melihat hasil score dari tiap fold yang dibentuk oleh model selection KFold.

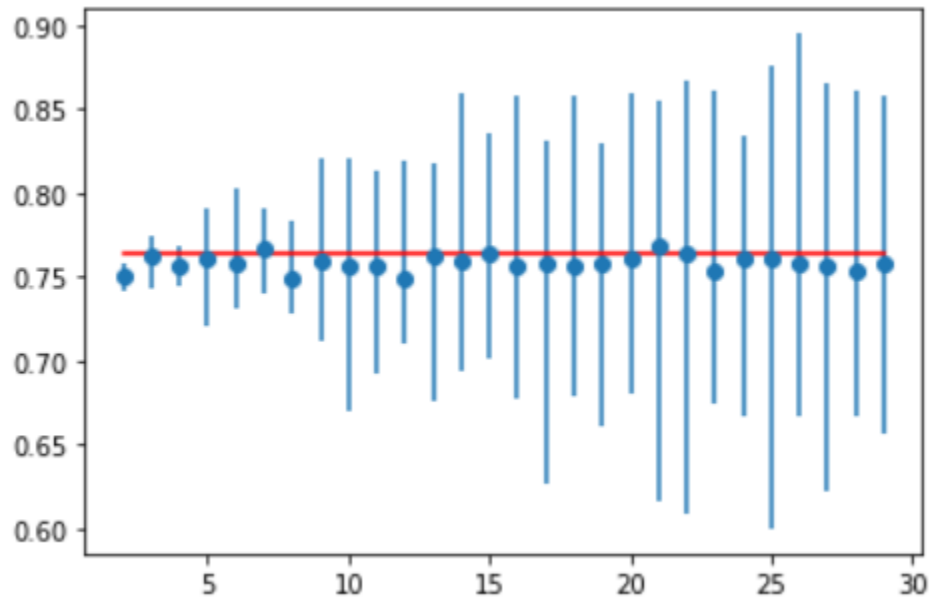
```
folds = range(2, 30)

means, mins, maxs = list(), list(), list()

for k in folds:
  cv = model_selection.KFold(n_splits=k, shuffle=True, random_state=1)
  k_mean, k_min, k_max = evaluate_model(cv)
  print('> fold=%d, accuracy=%.3f (%.3f,%.3f)' % (k, k_mean, k_min, k_max))
  means.append(k_mean)
  mins.append(k_mean - k_min)
  maxs.append(k_max - k_mean)
```

Setelah itu akan dibuat plotting error bar agar melihat perbandingan antara mean ideal yang sudah didapat pada saat menggunakan LeaveOneOut(), dan looping yang sudah dibuat.

```
pyplot.errorbar(folds, means, yerr=[mins, maxs], fmt='o')
pyplot.plot(folds, [ideal for _ in range(len(folds))], color='r')
pyplot.show()
```



Pada error bar, terlihat bahwa fold 15 adalah fold yang mempunyai nilai mean yang sama dengan nilai ideal(garis merah). Sehingga ketika dicek accuracy dari model,

```
# Cek model akurasi dengan kfold
kfold = model_selection.KFold(n_splits=15, random_state=seed[1], shuffle=True)
scores = cross_val_score(model, x, y, scoring='accuracy', cv=kfold, n_jobs=-1)
print('Accuracy: %.3f (Standard Deviation: %.3f)' % (np.mean(scores), np.std(scores)))
```

Akan didapatkan akurasi sebesar 76.5 % dengan standar deviasi sebesar 0.033..

```
Accuracy: 0.765 (Standard Deviation: 0.033)
```

Perlu diketahui bahwa algoritma untuk k-fold yang dipakai bersifat stokastik, sehingga hasil akurasi dapat berubah-ubah setiap kali kita menjalankan kodenya.

BAB V

EKSPERIMEN

Karena parameter - parameter sebelumnya yang dimasukkan ke model belum optimal, akan dicari nilai - nilai yang paling optimal berdasarkan list parameter yang disebutkan di bab pemodelan. Mencari nilai optimal akan menggunakan library GridSearchCV.

```
# Parameter
seed = [1301204243, 1301202398, 1301201586, 1301200457]
criterion = ['entropy', 'gini']
max_depth = [20, 40, 60, 70, 80, 100, None]
max_features = [2,4,6,8,10,12,14, 'auto']
n_estimators = [10, 30, 50, 80, 100, 150, 180]
```

Pertama, akan dicari parameter optimal pada bagian pembentukan Decision Tree Classifier.

```
# Mencari optimal parameter pada decision tree
d_parameters = {'criterion':criterion, 'max_depth':max_depth, 'max_features':max_features, 'random_state':seed}
d_grid = model_selection.GridSearchCV(DecisionTreeClassifier(), param_grid=d_parameters, verbose=False, cv=kfold, n_jobs=-1)
d_grid.fit(x, y)
```

Setelah itu, akan dicari parameter optimal di model Bagging Classifier.

```
# Mencari optimal parameter pada bagging classifier
cv = model_selection.StratifiedShuffleSplit(n_splits=15, test_size=.20, random_state=1)
parameters = {'n_estimators':n_estimators}
grid = model_selection.GridSearchCV(BaggingClassifier(base_estimator=d_grid.best_estimator_, bootstrap_features=False), param_grid=parameters, cv=cv,n_jobs=-1)
grid.fit(x,y)
```

Berdasarkan dari algoritma diatas, maka akan didapatkan hasil best score, best parameter, dan algoritma estimator yang sudah di optimasi berdasarkan parameternya.

```
--------Best Decision Tree Model--------
0.7131011608623549
{'criterion': 'entropy', 'max_depth': 40, 'max_features': 6, 'random_state': 1301201586}
DecisionTreeClassifier(criterion='entropy', max_depth=40, max_features=6,
                       random_state=1301201586)
--------Best Bagging Classifier Model--------
0.7746666666666666
{'n_estimators': 30}
BaggingClassifier(base_estimator=DecisionTreeClassifier(criterion='entropy',
                                                        max_depth=40,
                                                        max_features=6,
                                                        random_state=1301201586),
                  n_estimators=30)
```

Jika di cek menggunakan algoritma cross_val_score untuk keseluruhan data diperoleh, akurasi model sebesar 76.8% dan standar deviasi sebesar 0.023

```
# Mecoba model dengan parameter terbaik hasil pencarian gridsearch terhadap semua data dan menggunakan kfold
kfold = model_selection.KFold(n_splits=15, random_state=seed[1], shuffle=True)
scores = cross_val_score(grid.best_estimator_, x, y, scoring='accuracy', cv=kfold, n_jobs=-1)
print('Accuracy: %.3f (Standard Deviation: %.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.768 (Standard Deviation: 0.023)
```

```
# Mencoba model dengan parameter terbaik hasil pencarian gridsearch terhadap data testing yang dipakai saat tahap pemodelan

# Parameter
seed = [1301204243, 1301202398, 1301201586, 1301200457]
criterion = ['entropy', 'gini']
max_depth = [20, 40, 60, 70, 80, 100, None]
max_features = [2,4,6,8,10,12,14, 'auto']
n_estimators = [10, 30, 50, 80, 100, 150, 180]

# Decision Tree
dtree = DecisionTreeClassifier(criterion = criterion[0],
                               max_depth = max_depth[1],
                               max_features = max_features[2],
                               random_state = seed[2])

# Bagging classifier
model = BaggingClassifier(base_estimator = dtree,
                          n_estimators = n_estimators[1],
                          random_state = seed[2])
model.fit(x_train, y_train)

# Hasil
y_predict = model.predict(x_test)
print('------------Predicted Y Test-----------')
print(y_predict)
print('-----------------Y Test----------------')
print(y_test.values)
# Akurasi model
print('--------------------------------------------------------------------------')
print(f'Akurasi Model Terbaik Hasil Dari GridSearchCV Terhadap Data Testing: {accuracy_score(y_test, y_predict)}')
```

```
------------Predicted Y Test-----------
[0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 1 0 1
 0 1 0 0 0 0 0 1 0 0 0 0 0 0]
-----------------Y Test----------------
[0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0
 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1
 0 0 0 1 0 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1
 0 0 0 1 0 0 0 1 1 0 0 0 1 1 1 1 0 0 1 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 0 1
 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0]
--------------------------------------------------------------------------
Akurasi Model Terbaik Hasil Dari GridSearchCV Terhadap Data Testing: 0.81
```

Jika dicek menggunakan hanya data testing yang sama seperti pada tahap pemodelan maka diperoleh, akurasi model sebesar 81%.

Perlu diketahui bahwa algoritma untuk cross_val_score dan grid search yang dipakai bersifat stokastik, sehingga hasil akurasi atau hasil running dapat berubah-ubah setiap kali kita menjalankan kodenya.

BAB VI

KESIMPULAN

Dari hasil pemodelan, evaluasi, dan eksperimen yang telah dilakukan, kami mendapatkan bahwa hasil di tiap model menghasilkan score yang berbeda pula. Hasil dari pemodelan pertama, dengan parameter jenis criterion: ‘entropy’, max_depth: 70, max_feature: 4, n_estimators: 30, dan random_state: 1301204243 dan dengan bantuan Confusion Matrix, kita mendapatkan data 138 yang berupa True Positive, 8 yang berupa False Positive, 35 yang berupa False Negative, dan 19 yang True Negative yang menghasilkan akurasi sebesar 78.5%.

Setelah kami evaluasi model tersebut menggunakan K-Fold Cross Validation dengan lipatan sebanyak 15, didapatkan akurasi sebesar 76.5% dan standar deviasinya 0.033.

Lalu kami melakukan eksperimen untuk mencari parameter optimal pada Decision Tree Classifier dan Bagging Classifier untuk mendapatkan model yang optimal sesuai parameter yang kita definisikan dengan bantuan GridSearchCV. Hasil dari eksperimen tersebut menghasilkan bahwa model terbaik adalah model bagging classifier dengan jumlah tree (n_estimators) sebanyak 30, kemudian model decision tree dengan fungsi split (criterion) entropy, tingkat kedalaman tree (max_depth) sebesar 40, banyak fitur yang digunakan (max_features) sebanyak 6 fitur, dan seed untuk keacakan model (random_state) adalah 1301201586 diperoleh:

- Akurasi model hasil dari library grid search untuk keseluruhan data menggunakan cross_val_score adalah 76,8% dengan standar deviasi 0.023.
- Akurasi model hasil dari library grid search untuk data testing yang dipakai pada tahap pemodelan diperoleh skor akurasi sebesar 81%

Perlu diketahui bahwa algoritma grid search dan k-fold ini bersifat stokastik sehingga hasil setiap run dapat berbeda-beda.

DAFTAR PUSTAKA

https://machinelearningmastery.com/how-to-configure-k-fold-cross-validation/

https://www.kaggle.com/code/masumrumi/a-statistical-analysis-ml-workflow-of-titanic#Part-5:-Feature-Engineering

https://www.knowledgehut.com/blog/data-science/bagging-and-random-forest-in-machine-learning

https://www.section.io/engineering-education/implementing-bagging-algorithms-in-python/

Link GCollab:

https://colab.research.google.com/drive/1X5LnP1llZT4kmDJswFH2DYbi8x_DtJYC?usp=sharing

Link video presentasi:

https://youtu.be/0GKVp31uO1U