Tugas Besar Pembelajaran Mesin Lanjut
“Create Your Own AutoML”

Dibuat oleh :
Ridho Nobelino S (1301194245)
Naufal Edy Purwantono Scudetto (1301190197)

S1 Informatika
Telkom University, Bandung
2021/2022

Daftar Isi **2**

Formulasi Masalah **3**

Persiapan Data **3**

Pemodelan **5**

Hasil dan Pembahasan **7**

Kesimpulan **7**

1. Formulasi Masalah

Dalam tugas ini kita akan membangun sebuah fungsi untuk memanggil suatu model machine learning yang dapat menentukan parameter yang optimal secara otomatis.

2. Persiapan Data

Pertama yang perlu dilakukan adalah mengimport library python yang diperlukan diantaranya “sklearn”, “niapy”,”pandas”,”matplotlib”, dan “seaborn”. Setelah itu kita bisa memulai membaca file yang diperlukan yaitu file data untuk training dan juga test dalam hal ini yaitu data boston training dan boston test. Untuk memastikan data training yang telah di read menggunakan library pandas dengan memanggil kembali nama inisiasi file yang telah dibaca.

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd

from niapy.problems import Problem
from niapy.task import OptimizationType, Task
from niapy.algorithms.modified import HybridBatAlgorithm
from IPython.display import Image
from sklearn.preprocessing import Normalizer

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import StandardScaler
```

```
boston_train = pd.read_excel("./boston_train.xlsx")
boston_test = pd.read_excel("./boston_test.xlsx")
boston_train.head()

```

Menghasilkan output data berikut :

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.40771	0.0	6.20	1	0.507	6.164	91.3	3.0480	8	307	17.4	395.24	21.46	21.7
1	19.60910	0.0	18.10	0	0.671	7.313	97.9	1.3163	24	666	20.2	396.90	13.44	15.0
2	6.71772	0.0	18.10	0	0.713	6.749	92.6	2.3236	24	666	20.2	0.32	17.44	13.4
3	1.51902	0.0	19.58	1	0.605	8.375	93.9	2.1620	5	403	14.7	388.45	3.32	50.0
4	9.59571	0.0	18.10	0	0.693	6.404	100.0	1.6390	24	666	20.2	376.11	20.31	12.1

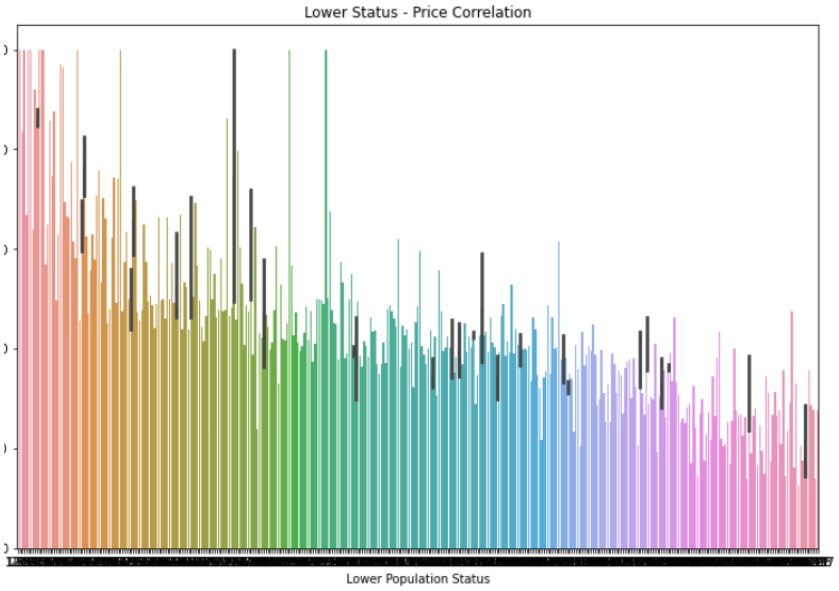
Tidak lupa juga menormalisasikan data agar memiliki rentang nilai yang sama menggunakan minmax scaller seperti pada code di bawah ini.

```
boston_train.isnull().sum()
scaler = StandardScaler()

x_train = boston_train[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']]
x_test = boston_test[['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT']]
y_train = boston_train[['PRICE']]
y_test = boston_test[['PRICE']]

x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)

print(len(x_train))
print(len(x_test))
```

Lalu dilakukan visualisasi data dalam bentuk barplot dan lineplot untuk melihat korelasi hungunan antara PRICE, RM, dan LSTAT. Dalam upaya mencoba untuk mengerti bagaimana mereka dapat saling mempengaruhi.

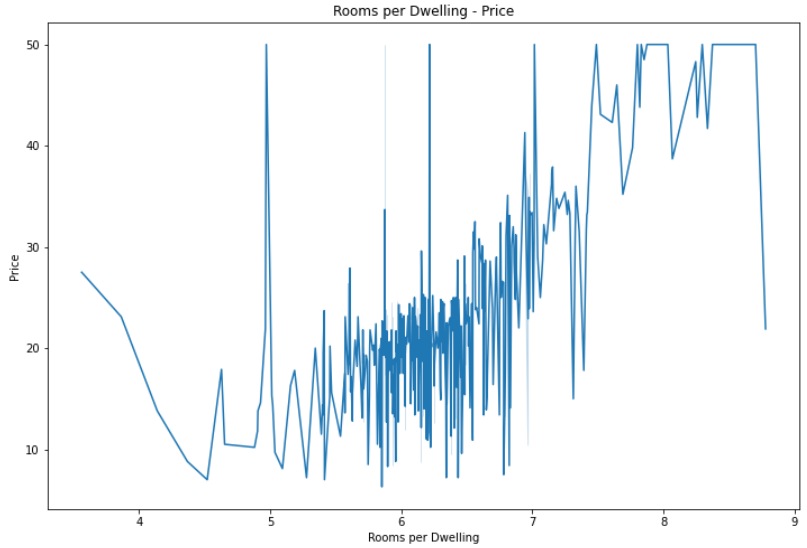
```
fig = plt.figure(figsize = (12,8))
boston = sns.barplot(x = 'LSTAT', y = 'PRICE', data = boston_train)
boston.set_title( "Lower Status - Price Correlation")

# This will add label to X-axis
boston.set_xlabel( "Lower Population Status")
# This will add label to Y-axis
boston.set_ylabel( "Price")
```

```
(0, 0.5, 'Price')
```



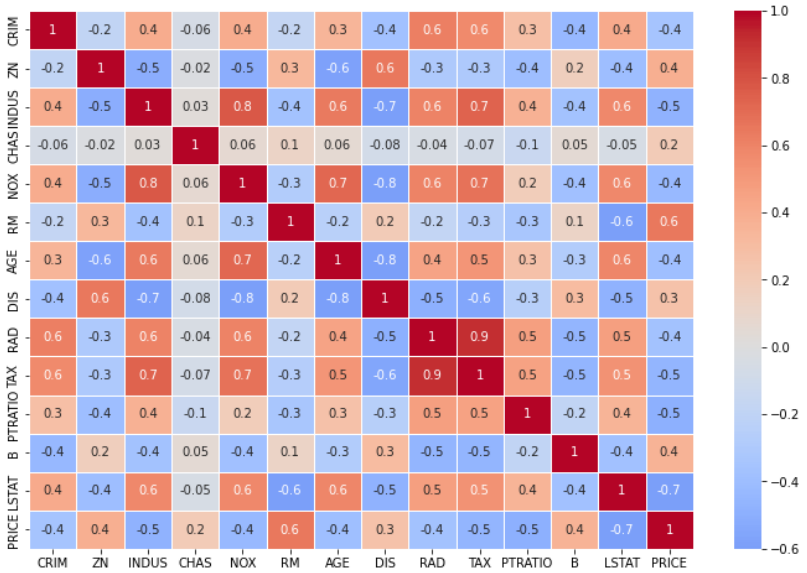

```
fig = plt.figure(figsize = (12,8))
boston = sns.lineplot(x = 'RM', y = 'PRICE', data = boston_train)
boston.set_title( "Rooms per Dwelling - Price")

# This will add label to X-axis
boston.set_xlabel( "Rooms per Dwelling")
# This will add label to Y-axis
boston.set_ylabel( "Price")
```

```
Text(0, 0.5, 'Price')
```



Selanjutnya akan dibikin heatmap untuk menentukan apa saja atribut yang berkorelasi positif dan negatif dengan harga.

```
corr_mat = boston_train.corr()
fig, ax = plt.subplots(figsize = (12, 8))
sns.heatmap(corr_mat, annot=True, ax=ax, fmt='.1g', cmap='coolwarm', linewidths=.5, vmin=-0.6, vmax=1, center= 0)
plt.show()
```



3. Pemodelan

Dalam tugas ini kita menggunakan metode Grey Wolf Optimization (GWO) untuk mendeteksi intrusi menggunakan Gray Wolf Optimization (GWO) dalam menyelesaikan masalah pemilihan fitur parameter yang optimal. Implementasinya didasarkan pada libsvm. Kompleksitas waktu kecocokan lebih dari sekadar kuadrat dengan jumlah sampel yang menyulitkan penskalaan ke kumpulan data dengan lebih dari beberapa 10.000 sampel.

```
#Code ini terinspirasi oleh link referral https://niapy.org/en/stable/tutorials/hyperparameter_optimization.html

#Memetakan vektor solusi kita ke parameter regresi
def hyperparametersGWO(x):
    degree= round(x[1] * 10 + 5, 2)
    kernels=('linear', 'rbf', 'sigmoid')
    kernel=kernels[int(x[2] * 2)]
    gammas=('scale', 'auto')
    gamma=gammas[int(x[3])]
    C= round(x[0] * 10 + 5, 2)

    parameters = {
        'degree': degree,
        'kernel' : kernel,
        'gamma': gamma,
        'C': C

    }

    return parameters

#Membangun regressor
def regressionGWO(x):
    parameters = hyperparametersGWO(x)
    return SVR(**parameters)

# Selanjutnya, kita perlu menulis kelas masalah khusus.
# Masalah tersebut akan menjadi 4 dimensi, dengan batas bawah dan atas masing-masing diatur ke 0 dan 1.
# Kelas juga akan menyimpan dataset pelatihan kami, di mana validasi silang 2 kali akan dilakukan.
# Fungsi fitness, yang akan kita maksimalkan, akan menjadi mean dari skor validasi silang.

class OptimizationGWO(Problem):
    def __init__(self, x_train, y_train):
        super().__init__(dimension=4, lower=0, upper=1)
        self.x_train = x_train
        self.y_train = y_train

    def _evaluate(self, x):
        model = regressionGWO(x)
        scores = cross_val_score(model, self.x_train, self.y_train, cv=2, n_jobs=-1)
        return scores.mean()
```

Dalam function def hyperparameter memanggil 4 parameter yaitu degree, kernel, gama dan c. Setelah mengembalikan parameter, selanjutnya parameter tersebut akan di panggil di regression function. Setelah itu kita membikin kondisi masalah khusus, masalah tersebut akan menggunakan 4 dimensi, dengan batas bawah dan atas masing-masing diatur ke 0 dan 1. Kelas juga akan menyimpan dataset pelatihan kami, di mana validasi silang 2 kali akan dilakukan. Fungsi fitness yang akan kita maksimalkan, akan menjadi mean dari skor validasi silang.

Setelah melakukan validasi silang dan juga menyimpan dataset class optimization akan digunakan untuk memulai eksperimen dengan menetapkan jumlah maksimum iterasi 150 dan mengatur ukuran populasi algoritma menjadi 15.

Eksperimen

```
# Sekarang saatnya menjalankan algoritma.
problem = OptimizationGWO(x_train, y_train)

# Kami menetapkan jumlah maksimum iterasi ke 150
task = Task(problem, max_iters=150, optimization_type=OptimizationType.MAXIMIZATION)

# Mengatur ukuran populasi algoritma menjadi 15
algorithm = GreyWolfOptimizer(population_size=15, seed=1234)
best_params_GWO, best_accuracy = algorithm.run(task)
```

Setelah menjalankan algoritma dan melakukan experimen, selanjutnya memanggil fungsi regression untuk kemudian memasukan hasil regresi dan menentukan nilai dari model.

Evaluasi

```
# Menjalankan regressor
default_model_GWO = SVR()
best_model_GWO = regressionGWO(best_params_GWO)

# Memasukkan hasil regresi
default_model_GWO.fit(x_train, y_train)
best_model_GWO.fit(x_train, y_train)

# Menentukan nilai model
default_score = default_model_GWO.score(x_test, y_test)
best_score_GWO = best_model_GWO.score(x_test, y_test)
```

Setelah didapat score terbaik akan disimpan dan menjadi output pada section berikutnya.

4. Hasil dan Pembahasan

Dalam hasil dari program ini dengan iterasi 150 dan ukuran populasi algoritma 15 maka nilai terbaik di dapatkan sebagai berikut :

```
# Menunjukkan akurasi dari setiap model
print('Akurasi Model Bawaan   :', default_score)
print('Akurasi Model Unggulan :', best_score_GWO)

#Menunjukkan parameter dan nilai terbaik
print('Parameter Terbaik      :', hyperparametersGWO(best_params_GWO))
print('Nilai Terbaik          :', best_score_GWO)
```

```
Akurasi Model Bawaan   : 0.6652442203874216
Akurasi Model Unggulan : 0.8574725501876372
Parameter Terbaik      : {'degree': 9.73, 'kernel': 'rbf', 'gamma': 'auto', 'C': 13.59}
Nilai Terbaik          : 0.8574725501876372
```

5. Kesimpulan

Pada tugas ini kita bertujuan untuk membangun sebuah fungsi untuk memanggil suatu model machine learning yang dapat menentukan parameter yang optimal secara otomatis. Dan kita memutuskan untuk menggunakan Grey Wolf Optimization. Algoritma Grey Wolf Optimization (GWO) adalah teknologi pengoptimalan meta-heuristik baru dimana prinsipnya adalah meniru perilaku serigala di alam dalam aspek berburu secara kooperatif. Grey Wolf Optimization memiliki kinerja yang baik untuk masalah optimasi yang solusi optimalnya adalah 0. Kemudian ditemukan lebih lanjut bahwa ketika GWO menyelesaikan fungsi optimasi yang sama, semakin jauh solusi optimal fungsi dari 0, semakin buruk kinerjanya.

Dari hasil eksperimen kita, terbukti bahwa Grey Wolf Optimization (GWO) menghasilkan akurasi yang lebih tinggi dibandingkan model bawaan. Ini dikarenakan ia mengoptimalisasikan pemilihan parameter dalam model. Maka dari itu kita dapat menyimpulkan bahwa telah terwujudnya tujuan awal kita.