

# PEMBELAJARAN MESIN DAN PEMBELAJARAN MENDALAM

# **Modul 3**

# **Supervised Learning**

Yohanes Sigit Purnomo Wuryo Putro, ST., M.Kom., Ph.D.  
Aloysius Gonzaga Pradnya Sidhwara, S.T., M.Eng.  
Atanasius Surya Gunadharma  
Kimmi Caitlyn Ariesta

# Tugas Modul Supervised Learning

Dalam rangka meningkatkan kualitas pendidikan di SMA Ayodya, tim akademik berupaya untuk memprediksi Indeks Prestasi Kumulatif (IPK) siswa berdasarkan nilai mereka dari berbagai mata pelajaran. Para guru dan pimpinan sekolah menyadari bahwa dengan prediksi IPK yang lebih baik, mereka dapat memberikan bimbingan dan dukungan lebih awal kepada siswa yang mungkin membutuhkan bantuan.

Untuk mencapai tujuan ini, tim akademik bekerja sama dengan tim data scientist. Data yang dikumpulkan dari siswa SMA Ayodya mencakup nilai dari beberapa mata pelajaran inti, seperti Matematika (MTK), Bahasa Inggris (ING), dan Bahasa Indonesia (BHS) dari semester 1 dan 2. Setiap mata pelajaran memiliki nilai Kriteria Ketuntasan Minimal (KKM) yang harus dicapai oleh siswa.

Setelah data dikumpulkan, tim data scientist diminta untuk melatih model machine learning yang dapat memprediksi IPK siswa berdasarkan nilai-nilai tersebut. Model ini akan diuji dengan data yang ada untuk melihat performanya dalam memprediksi nilai IPK.

Instruksi dari pimpinan tim data scientist adalah sebagai berikut:

Selamat datang untuk rekan-rekan baru tim data scientist. Berikut adalah instruksi yang dapat saya berikan untuk membuat model machine learning menggunakan algoritma Ridge Regression, Lasso, Support Vector Regression (SVR), dan Gradient Boosting Regression (GBR):

1. Pahami dataset: Sebelum memulai pembuatan model, pastikan Anda memahami dataset yang akan digunakan. Periksa apakah terdapat missing values
2. Data preprocessing: Anda harus memastikan tidak ada data yang kosong atau null pada kolom fitur. Untuk nilai parameter **random\_state** pada pemisahan dataset disesuaikan dengan dua digit terakhir nomor pegawai Anda (**random\_state: dua digit terakhir NPM**)
3. Buat model Ridge Regression: Gunakan library scikit-learn untuk membuat model Ridge Regression. Anda diperbolehkan untuk menyesuaikan parameter `reg_alpha` dan `feature_selection_k` agar model dapat bekerja dengan baik pada dataset
4. Buat model Lasso Regression: Gunakan library scikit-learn untuk membuat model Lasso Regression. Anda diperbolehkan untuk menyesuaikan parameter `reg_alpha` dan `feature_selection_k` agar model dapat bekerja dengan baik pada dataset
5. Buat model Support Vector Regression (SVR): Gunakan library scikit-learn untuk membuat model SVR. Anda diperbolehkan untuk menyesuaikan parameter `C`, `epsilon`, dan `feature_selection_k` agar model dapat bekerja dengan baik pada dataset
6. Buat model Gradient Boosting Regressor (GBR): Gunakan library scikit-learn untuk membuat model GBR. Anda diperbolehkan untuk menyesuaikan parameter `n_estimators`, `learning_rate`, `max_depth`, dan `feature_selection_k` agar model dapat bekerja dengan baik pada dataset
7. Evaluasi model: Setelah membuat semua model, lakukan evaluasi untuk mengetahui performa masing-masing model
8. Buat interface streamlit: Setelah model tersimpan, buat antarmuka yang dapat digunakan oleh tim akademik untuk memprediksi IPK

## LOAD DATASET

- Tahap pertama adalah load dataset berdasarkan path di mana dataset disimpan
- Karena dataset dalam bentuk file CSV, maka kita menggunakan fungsi `read_csv` dari pandas

```
In [33]: import pandas as pd  
import numpy as np
```

```
#Load dataset menggunakan function read_csv dari pandas  
df_ipk = pd.read_csv(r'C:\Users\HP\Downloads\dataset_regresi_IPK.csv') #sesuaikan dengan Lokasi file dataset tempat anda menyimpannya  
df_ipk.head(10)
```

Out[33]:

	name	1-1-mtk	1-2-mtk	2-1-mtk	2-2-mtk	1-1-ing	1-2-ing	2-1-ing	2-2-ing	1-1-bhs	...	2-2-kkm_bhs	1-1-kkm_ing	1-2-kkm_ing	2-1-kkm_ing	2-2-kkm_ing	1-1-kkm_mtk	1-2-kkm_mtk	2-1-kkm_mtk	2-2-kkm_mtk	IPK
0	student1	93.0	84.0	93.0	91.0	85.0	94.0	85.0	94.0	91.0	...	78.0	75.0	75.0	78.0	78.0	75.0	75.0	78.0	78.0	4.000000
1	student2	80.0	82.0	82.0	84.0	80.0	84.0	85.0	83.0	83.0	...	78.0	75.0	75.0	78.0	78.0	75.0	75.0	78.0	78.0	0.000000
2	student3	71.0	80.0	96.0	85.0	82.0	84.0	86.0	80.0	84.0	...	70.0	70.0	70.0	70.0	70.0	70.0	70.0	70.0	70.0	3.295000
3	student4	81.0	82.0	87.0	82.0	81.0	86.0	82.0	84.0	87.0	...	78.0	75.0	75.0	78.0	78.0	75.0	75.0	78.0	78.0	3.685000
4	student5	83.0	81.0	88.0	84.0	80.0	88.0	83.0	85.0	86.0	...	78.0	75.0	75.0	78.0	78.0	75.0	75.0	78.0	78.0	3.752500
5	student6	88.0	78.0	77.0	80.0	80.0	80.0	77.0	79.0	78.0	...	67.0	67.0	67.0	67.0	67.0	67.0	67.0	67.0	67.0	3.740909
6	student7	89.0	89.0	88.0	89.0	85.0	90.0	80.0	82.0	79.0	...	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	3.052380
7	student8	78.0	75.0	78.0	82.0	77.0	87.0	88.0	80.0	84.0	...	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	75.0	1.585000
8	student9	93.0	86.0	93.0	76.0	77.0	81.0	74.0	84.0	89.0	...	72.0	72.0	72.0	72.0	72.0	72.0	72.0	72.0	72.0	3.547619
9	student10	73.0	75.0	68.0	78.0	83.0	82.0	73.0	73.0	75.0	...	73.0	73.0	73.0	73.0	73.0	73.0	73.0	73.0	73.0	3.007692

10 rows × 26 columns

## DATA CLEANSING

- drop fitur yang tidak relevan dengan pelatihan model

```
In [34]: df_ipk2 = df_ipk.drop(['name'], axis=1)  
df_ipk2.head()
```

Out[34]:

	1-1-mtk	1-2-mtk	2-1-mtk	2-2-mtk	1-1-ing	1-2-ing	2-1-ing	2-2-ing	1-1-bhs	...	2-2-kkm_bhs	1-1-kkm_ing	1-2-kkm_ing	2-1-kkm_ing	2-2-kkm_ing	1-1-kkm_mtk	1-2-kkm_mtk	2-1-kkm_mtk	2-2-kkm_mtk	IPK	
0	93.0	84.0	93.0	91.0	85.0	94.0	85.0	94.0	91.0	...	78.0	75.0	75.0	78.0	78.0	75.0	75.0	78.0	78.0	4.000000	
1	80.0	82.0	82.0	84.0	80.0	84.0	85.0	83.0	83.0	82.0	...	78.0	75.0	75.0	78.0	78.0	75.0	75.0	78.0	78.0	0.000000
2	71.0	80.0	96.0	85.0	82.0	84.0	86.0	80.0	84.0	86.0	...	70.0	70.0	70.0	70.0	70.0	70.0	70.0	70.0	70.0	3.295000
3	81.0	82.0	87.0	82.0	81.0	86.0	82.0	84.0	87.0	84.0	...	78.0	75.0	75.0	78.0	78.0	75.0	75.0	78.0	78.0	3.685000
4	83.0	81.0	88.0	84.0	80.0	88.0	83.0	85.0	86.0	85.0	...	78.0	75.0	75.0	78.0	78.0	75.0	75.0	78.0	78.0	3.752500

5 rows × 25 columns

No Copy

No Copy

Watermarkly

```
In [35]: #cek tipe data  
df_ipk2.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 778 entries, 0 to 777  
Data columns (total 25 columns):  
 #   Column      Non-Null Count  Dtype     
---  --          --          --         
 0   1-1-mtk    778 non-null    float64  
 1   1-2-mtk    778 non-null    float64  
 2   2-1-mtk    778 non-null    float64  
 3   2-2-mtk    778 non-null    float64  
 4   1-1-ing    778 non-null    float64  
 5   1-2-ing    778 non-null    float64  
 6   2-1-ing    778 non-null    float64  
 7   2-2-ing    778 non-null    float64  
 8   1-1-bhs    778 non-null    float64  
 9   1-2-bhs    778 non-null    float64  
 10  2-1-bhs    778 non-null    float64  
 11  2-2-bhs    778 non-null    float64  
 12  1-1-kkm_bhs 778 non-null  float64  
 13  1-2-kkm_bhs 778 non-null  float64  
 14  2-1-kkm_bhs 778 non-null  float64  
 15  2-2-kkm_bhs 778 non-null  float64  
 16  1-1-kkm_ing 778 non-null  float64  
 17  1-2-kkm_ing 778 non-null  float64  
 18  2-1-kkm_ing 778 non-null  float64  
 19  2-2-kkm_ing 778 non-null  float64  
 20  1-1-kkm_mtk 778 non-null  float64  
 21  1-2-kkm_mtk 778 non-null  float64  
 22  2-1-kkm_mtk 778 non-null  float64  
 23  2-2-kkm_mtk 778 non-null  float64  
 24  IPK        762 non-null    float64  
dtypes: float64(25)  
memory usage: 152.1 KB
```

- Dataset dapat kita deskripsikan secara statistik menggunakan fungsi `describe()` dan menghasilkan luaran seperti berikut:

```
In [36]: #cek deskripsi data  
df_ipk2.describe()
```

	1-1-mtk	1-2-mtk	2-1-mtk	2-2-mtk	1-1-ing	1-2-ing	2-1-ing	2-2-ing	1-1-bhs	1-2-bhs	...	2-2-kkm_bhs	1-1-kkm_ing	1-2-kkm_ing	2-1-kkm_ing	2-2-kkm_ing	1-1-kkm_mtk	kki
count	778.000000	778.000000	778.000000	778.000000	778.000000	778.000000	778.000000	778.000000	778.000000	778.000000	...	778.000000	778.000000	778.000000	778.000000	778.000000	778.000000	778.000000
mean	81.075527	82.555270	84.313805	83.985437	83.085681	84.486928	84.490578	85.011568	82.783779	84.286131	...	73.115681	72.231362	72.223650	73.065553	72.172237	72.1	
std	6.203185	5.999447	5.900668	5.889155	5.698347	5.658098	5.378797	5.394704	5.516622	5.316999	...	3.303909	2.881252	2.877393	3.309760	3.307037	2.904512	2.9
min	60.000000	60.000000	68.000000	66.000000	64.000000	62.000000	68.000000	68.000000	67.000000	70.000000	...	67.000000	67.000000	67.000000	67.000000	67.000000	67.000000	67.000000
25%	77.000000	78.000000	80.000000	80.000000	79.000000	81.000000	81.000000	81.250000	79.000000	81.000000	...	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000	70.000000
50%	81.000000	82.875000	85.000000	84.000000	83.000000	84.000000	85.000000	85.000000	83.000000	84.000000	...	75.000000	73.000000	73.000000	73.500000	73.500000	73.000000	73.000000
75%	85.000000	86.000000	88.000000	88.000000	87.000000	88.000000	88.000000	89.000000	87.000000	88.000000	...	75.000000	75.000000	75.000000	75.000000	75.000000	75.000000	75.000000
max	100.000000	100.000000	99.000000	100.000000	98.000000	99.000000	99.000000	99.000000	98.000000	99.000000	...	79.000000	78.000000	78.000000	79.000000	79.000000	78.000000	78.000000

8 rows × 25 columns

```
In [37]: #cek distribusi data  
print(df_ipk2['IPK'].value_counts())
```

IPK	Count
0.000000	26
4.000000	5
3.979069	4
3.706976	4
3.300000	4
..	
2.131707	1
3.402272	1
3.865909	1
2.285714	1
3.569047	1
Name: IPK, Length: 609, dtype: int64	

- Data yang hilang akan membuat model menjadi error, untuk itu kita perlu menanganinya
- Kita dapat melakukan pengecekan pada data apabila ada data yang bernilai null, kosong, atau Nan dengan cara seperti berikut:

No Copy



Watermarkly

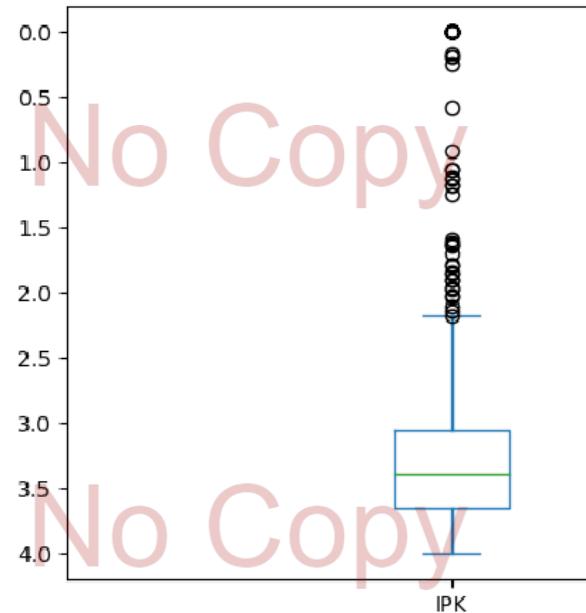


## DELETE DATA KOSONG

- Gunakan fungsi plot dengan parameter `kind='box'` untuk melihat sebaran data dari fitur "IPK"

In [39]:

```
import matplotlib.pyplot as plt  
  
df_ipk2.IPK.plot(kind='box')  
plt.gca().invert_yaxis()  
plt.show()
```



- Selanjutnya buatlah sebuah fungsi yang akan digunakan untuk pembersihan outlier dengan metode inter-quartile range
- Selanjutnya tampilkan perbandingan jumlah baris dari Dataframe sebelum dan sesudah pembersihan outlier
- Gunakan fungsi plot dengan parameter `kind='box'` untuk melihat sebaran data dari fitur IPK setelah outlier dibersihkan

```
In [40]: #menggunakan fungsi remove_outlier untuk menghilangkan outlier
from pandas.api.types import is_numeric_dtype
def remove_outlier(df_in):
    for col_name in list(df_in.columns):
        if is_numeric_dtype(df_in[col_name]):
            q1 = df_in[col_name].quantile(0.25)
            q3 = df_in[col_name].quantile(0.75)

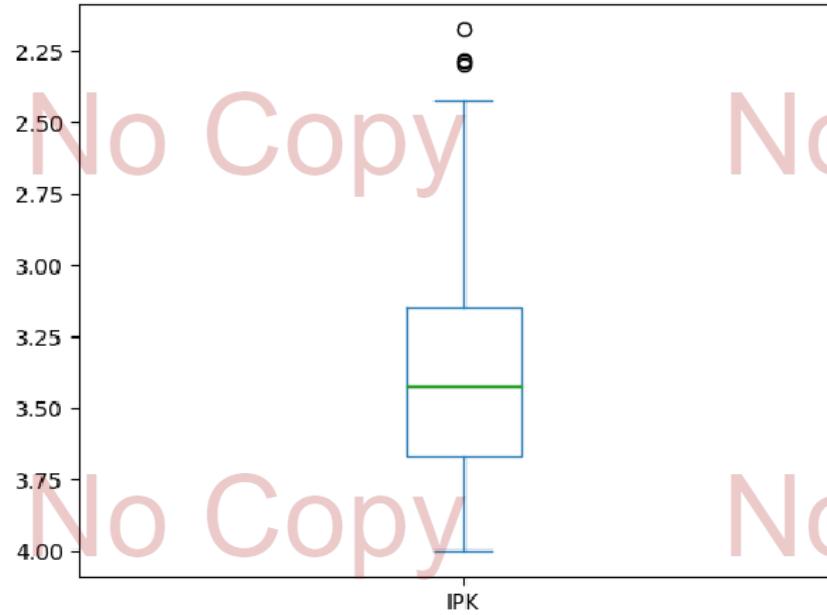
            iqr = q3-q1
            batas_atas = q3 + (1.5 * iqr)
            batas_bawah = q1 - (1.5 * iqr)

            df_out = df_in.loc[(df_in[col_name] >= batas_bawah) & (df_in[col_name] <= batas_atas)]
    return df_out

df_ipk_clean = remove_outlier(df_ipk2)
print("Jumlah baris DataFrame sebelum dibuang outlier", df_ipk2.shape[0])
print("Jumlah baris DataFrame sesudah dibuang outlier", df_ipk_clean.shape[0])
df_ipk_clean.IPK.plot(kind='box', vert=True)

#untuk membalik sumbu y
plt.gca().invert_yaxis()
plt.show()
```

Jumlah baris DataFrame sebelum dibuang outlier 778  
Jumlah baris DataFrame sesudah dibuang outlier 705



- Mengecek ulang apakah masih ada data yang kosong, null, atau NaN



## Train-test split

- Lakukan pemisahan antara data fitur dengan data target
- Selanjutnya, lakukan train-test split dengan menggunakan rasio parameter test\_size dan train\_size sebesar 25:75 serta random\_state dengan nilai 42

```
In [45]: #Menggunakan fungsi train_test_split untuk membagi data train dan test
from sklearn.model_selection import train_test_split
```

```
X_regress = df_ipk_clean.drop('IPK' ,axis=1)
y_regress = df_ipk_clean.IPK

X_train_ipk, X_test_ipk, y_train_ipk, y_test_ipk = train_test_split(X_regress, y_regress, \
                                                               test_size=0.25,
                                                               random_state=42) #ubah menjadi 2 digit NPM
```

## Ridge Regressor

- Buatlah pipeline yang terdiri dari data scaling dan regressor
- Buatlah parameter grid untuk parameter algoritme Ridge
- Buatlah GSCV yang digunakan untuk mencari model terbaik dengan metode validasi 5-fold CV
- Tampilkan best model, nilai koefisien yang diperoleh dengan menggunakan atribut coef dan nilai intercept dari persamaan linier dengan menggunakan atribut intercept
- Tampilkan Mean Absolute Error, Mean Squared Error, dan Root Mean Squared Error

No Copy

No Copy Watermarkly

```
In [46]: #Import Library yang diperlukan untuk model ridge regression
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error #Digunakan untuk menghitung MSE dan MAE yang berguna untuk evaluasi model
```

```
#Mendefinisikan parameter grid untuk RR
pipe_Ridge = Pipeline(steps=[
    ('scale', StandardScaler()),
    ('feature_selection', SelectKBest(score_func=f_regression)), #Seleksi fitur menggunakan SelectKBest dengan f_regression sebagai fungsi skor
    ('reg', Ridge())
])
```

```
#Mendefinisikan parameter
param_grid_Ridge = {
    'reg_alpha': [0.01, 0.1, 1, 10, 100], #Nilai aplha berguna untuk mengontrol tingkat regularisasi
    'feature_selection_k': np.arange(1, 20)
}
```

```
#Mengimplementasikan GridSearch untuk RR
GSCV_RR = GridSearchCV(pipe_Ridge, param_grid_Ridge, cv=5,
                       scoring='neg_mean_squared_error', error_score='raise') #'neg_mean_squared_error' digunakan untuk menghitung MSE,
#semakin kecil MSE, MAE, dan RMSE, maka semakin baik juga modelnya
GSCV_RR.fit(X_train_ipk, y_train_ipk)
```

```
# Mencetak model terbaik dan parameter terbaik
print("Best model:{}".format(GSCV_RR.best_estimator_))
print("Ridge best parameters:{}".format(GSCV_RR.best_params_))
# Mengakses koefisien dan intercept dari model terbaik
print("Koefisien/bobot:{}".format(GSCV_RR.best_estimator_.named_steps['reg'].coef_))
print("Intercept/bias:{}".format(GSCV_RR.best_estimator_.named_steps['reg'].intercept_))
```

```
# Buat Prediksi
Ridge_predict = GSCV_RR.predict(X_test_ipk)

mse_Ridge = mean_squared_error(y_test_ipk, Ridge_predict)
mae_Ridge = mean_absolute_error(y_test_ipk, Ridge_predict)
```

```
# Menghitung dan mencetak metrik evaluasi
print("Ridge Mean Squared Error (MSE): {}".format(mse_Ridge))
print("Ridge Mean Absolute Error (MAE): {}".format(mae_Ridge))
print("Ridge Root Mean Squared Error: {}".format(np.sqrt(mse_Ridge)))
```

```
Best model:Pipeline(steps=[('scale', StandardScaler()),
                           ('feature_selection',
                            SelectKBest(k=15,
                                        score_func=<function f_regression at 0x000001D915F3C220>)),
                           ('reg', Ridge(alpha=100))])
Ridge best parameters:{'feature_selection_k': 15, 'reg_alpha': 100}
Koefisien/bobot:[ 0.01801446  0.01284599  0.03029103  0.02048213  0.00268661  0.05192312
  0.02700493  0.03051754  0.00800545  0.00041879  0.00529828  0.02853707
 -0.00706174 -0.00573325 -0.01606943]
Intercept/bias:3.3776050776515154
Ridge Mean Squared Error (MSE): 0.09072229930844042
Ridge Mean Absolute Error (MAE): 0.23833548926056355
Ridge Root Mean Squared Error: 0.3012014264714568
```

- Tambahkan hasil prediksi model Ridge Regression pada dataframe hasil

No Copy

No Copy

No Copy

No Copy

No Copy

No Copy

Watermarkly

In [60]: #Membuat data frame untuk menampung hasil prediksi

```
df_results = pd.DataFrame(y_test_ipk, columns=['IPK']) # Menyimpan nilai IPK asli
df_results = pd.DataFrame(y_test_ipk) # Menyimpan nilai hasil prediksi Ridge
df_results['Ridge Prediction'] = Ridge_predict

# Menambahkan kolom yang berisi selisih antara IPK prediksi dan IPK asli
df_results['Selisih_IPK_RR'] = df_results['Ridge Prediction'] - df_results['IPK']

df_results.head()
```

Out[60]:

	IPK	Ridge Prediction	Selisih_IPK_RR
517	3.292500	3.338426	0.045926
87	3.423255	3.442874	0.019619
82	3.582500	3.475864	-0.106636
228	2.778571	3.038370	0.259799
349	3.516666	3.159569	-0.357097

In [61]: #Cek deskripsi data  
df\_results.describe()

Out[61]:

	IPK	Ridge Prediction	Selisih_IPK_RR
count	177.000000	177.000000	177.000000
mean	3.392731	3.393127	0.000396
std	0.325741	0.153206	0.302056
min	2.509090	3.038370	-0.577734
25%	3.175000	3.284496	-0.201274
50%	3.423255	3.391923	-0.043074
75%	3.639473	3.488273	0.174394
max	4.000000	3.850325	0.969770

## Lasso Regression

- Buatlah pipeline yang terdiri dari data scaling dan regressor
- Buatlah parameter grid untuk parameter algoritme Lasso
- Buatlah GSCV yang digunakan untuk mencari model terbaik dengan metode validasi 5-fold CV
- Tampilkan best model, nilai koefisien yang diperoleh dengan menggunakan atribut coef dan nilai intercept dari persamaan linier dengan menggunakan atribut intercept
- Tampilkan Mean Absolute Error, Mean Squared Error, dan Root Mean Squared Error

No Copy

No Copy

No Copy

No Copy

Watermarkly

```
In [57]: #import library yang diperlukan untuk model ridge regression
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error #Digunakan untuk menghitung MSE dan MAE yang berguna untuk evaluasi model
# Define the pipeline for LR
pipe_Lasso = Pipeline(steps=[
    ('scale', StandardScaler()),
    ('feature_selection', SelectKBest(score_func=f_regression)), #Seleksi fitur menggunakan SelectKBest dengan f_regression sebagai fungsi skor
    ('reg', Lasso(max_iter=1000)) #max_iter digunakan untuk menentukan batas iterasi
])

#Mendefinisikan parameter grid untuk LR
param_grid_Lasso = {
    'reg_alpha': [0.01, 0.1, 1, 10, 100], #Nilai aplha berguna untuk mengontrol tingkat regularisasi
    'feature_selection_k': np.arange(1, 20)
}

#Mengimplementasikan GridSearchCV untuk LR
GSCV_Lasso = GridSearchCV(pipe_Lasso, param_grid_Lasso, cv=5, scoring='neg_mean_squared_error') #'neg_mean_squared_error' digunakan untuk menghitung MSE,
#semakin kecil MSE, MAE, dan RMSE, maka semakin baik juga modelnya
GSCV_Lasso.fit(X_train_ipk, y_train_ipk)

#Mencetak model terbaik dan parameter terbaik
print("Best model:{}".format(GSCV_Lasso.best_estimator_))
print("Lasso best parameters:{}".format(GSCV_Lasso.best_params_))
#Mengakses koefisien dan intercept dari model terbaik
print("Koefisien/bobot:{}".format(GSCV_Lasso.best_estimator_.named_steps['reg'].coef_))
print("Intercept/bias:{}".format(GSCV_Lasso.best_estimator_.named_steps['reg'].intercept_))

#Membuat Prediksi
Lasso_predict = GSCV_Lasso.predict(X_test_ipk)

#Menghitung dan mencetak metrik evaluasi
mse_Lasso = mean_squared_error(y_test_ipk, Lasso_predict)
mae_Lasso = mean_absolute_error(y_test_ipk, Lasso_predict)

print("Lasso Mean Squared Error (MSE): {}".format(mse_Lasso))
print("Lasso Mean Absolute Error (MAE): {}".format(mae_Lasso))
print("Lasso Root Mean Squared Error: {}".format(np.sqrt(mse_Lasso)))
```

```
Best model:Pipeline(steps=[('scale', StandardScaler()),
                           ('feature_selection',
                            SelectKBest(k=15,
                                         score_func=<function f_regression at 0x000001D915F3C220>)),
                           ('reg', Lasso(alpha=0.01))])
Lasso best parameters:{'feature_selection_k': 15, 'reg_alpha': 0.01}
Koefisien/bobot:[ 0.01488563  0.01002612  0.0310264   0.01835883  0.          0.06133505
  0.02425218  0.0311455   0.          0.          0.02928967
 -0.          -0.          -0.01796958]
Intercept/bias:3.3776050776515154
Lasso Mean Squared Error (MSE): 0.09086828784352101
Lasso Mean Absolute Error (MAE): 0.24013325069277436
Lasso Root Mean Squared Error: 0.3014436727541665
```

- Tambahkan hasil prediksi model Lasso Regression pada dataframe hasil

In [62]: #Menambahkan prediksi ke dalam data frame hasil

```
df_results['Lasso Prediction'] = Lasso_predict
df_results = pd.DataFrame(y_test_ipk) # Menyimpan nilai hasil prediksi Lasso
df_results['Lasso Prediction'] = Lasso_predict

# Menambahkan kolom yang berisi selisih antara IPK prediksi dan IPK asli
df_results['Selisih_IPK_LR'] = df_results['Lasso Prediction'] - df_results['IPK'] #Prediksi IPK - IPK asli
df_results.head()
```

Out[62]:

	IPK	Lasso Prediction	Selisih_IPK_LR
517	3.292500	3.344446	0.051946
87	3.423255	3.410330	-0.012925
82	3.582500	3.466212	-0.116288
228	2.778571	3.038451	0.259880
349	3.516666	3.165573	-0.351093

In [74]: #Cek deskripsi data

```
df_results.describe()
```

Out[74]:

	IPK	Ridge Prediction	Selisih Ridge	Lasso Prediction	Selisih Lasso
count	177.000000	177.000000	177.000000	177.000000	177.000000
mean	3.392731	3.393127	-0.000396	3.392159	0.000572
std	0.325741	0.153206	0.302056	0.146426	0.302298
min	2.509090	3.038370	-0.969770	3.034900	-0.934173
25%	3.175000	3.284496	-0.174394	3.285882	-0.167788
50%	3.423255	3.391923	0.043074	3.382823	0.041176
75%	3.639473	3.488273	0.201274	3.482827	0.194580
max	4.000000	3.850325	0.577734	3.814728	0.570863

## SVR (Support Vector Regressor)

- Buatlah pipeline yang terdiri dari data scaling dan regressor
- Buatlah parameter grid untuk parameter algoritme SVR
- Buatlah GSCV yang digunakan untuk mencari model terbaik dengan metode validasi 5-fold CV
- Tampilkan best model, nilai koefisien yang diperoleh dengan menggunakan atribut coef dan nilai intercept dari persamaan linier dengan menggunakan atribut intercept
- Tampilkan Mean Absolute Error, Mean Squared Error, dan Root Mean Squared Error

No Copy

No Copy

Watermarkly

```
In [64]: #import library yang diperlukan untuk model ridge regression
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error #Digunakan untuk menghitung MSE dan MAE yang berguna untuk evaluasi model
```

```
# Define the pipeline for SVR
pipe_SVR = Pipeline(steps=[
    ('scale', StandardScaler()),
    ('feature_selection', SelectKBest(score_func=f_regression)), #Seleksi fitur menggunakan SelectKBest dengan f_regression sebagai fungsi skor
    ('reg', SVR(kernel='linear')) #Menerapkan model SVR dengan kernel linear
])
```

```
#Mendefinisikan parameter grid untuk SVR
param_grid_SVR = {
    'reg_C': [0.01, 0.1, 1, 10, 100],
    'reg_epsilon': [0.1, 0.2, 0.5, 1], #Toleransi eror terbaik
    'feature_selection_k': np.arange(1,20)
}

#Mngimplementasikan GridSearchCV untuk SVR
GSCV_SVR = GridSearchCV(pipe_SVR, param_grid_SVR, cv=5, scoring='neg_mean_squared_error') #'neg_mean_squared_error' digunakan untuk menghitung MSE,
#semakin kecil MSE, MAE, dan RMSE, maka semakin baik juga modelnya
GSCV_SVR.fit(X_train_ipk, y_train_ipk)
```

```
#Mencetak model terbaik dan parameter terbaik
print("Best model:{}".format(GSCV_SVR.best_estimator_))
print("SVR best parameters:{}".format(GSCV_SVR.best_params_))
#Mengakses koefisien dan intercept dari model terbaik
print("Koefisien/bobot:{}".format(GSCV_SVR.best_estimator_.named_steps['reg'].coef_))
print("Intercept/bias:{}".format(GSCV_SVR.best_estimator_.named_steps['reg'].intercept_))

#Membuat Prediksi
SVR_predict = GSCV_SVR.predict(X_test_ipk)
```

```
#Menghitung dan mencetak metrik evaluasi
mse_SVR = mean_squared_error(y_test_ipk, SVR_predict)
mae_SVR = mean_absolute_error(y_test_ipk, SVR_predict)

print("SVR Mean Squared Error (MSE): {}".format(mse_SVR))
print("SVR Mean Absolute Error (MAE): {}".format(mae_SVR))
print("SVR Root Mean Squared Error: {}".format(np.sqrt(mse_SVR)))
```

```
Best model:Pipeline(steps=[('scale', StandardScaler()),
                           ('feature_selection',
                            SelectKBest(k=7,
                                         score_func=<function f_regression at 0x000001D915F3C220>)),
                           ('reg', SVR(C=0.01, epsilon=0.2, kernel='linear'))])
SVR best parameters:{'feature_selection_k': 7, 'reg_C': 0.01, 'reg_epsilon': 0.2}
Koefisien/bobot:[[0.01913641 0.0209015  0.02816713 0.06011631 0.03056364 0.03394113
  0.04291088]]
Intercept/bias:[3.39784976]
SVR Mean Squared Error (MSE): 0.09185806582442452
SVR Mean Absolute Error (MAE): 0.23851169425571525
SVR Root Mean Squared Error: 0.3030809558920265
```

- Tambahkan hasil prediksi model SVR pada dataframe hasil



In [75]: #Menambahkan prediksi ke dalam data frame hasil

```
df_results['SVR Prediction'] = SVR_predict
df_results = pd.DataFrame(y_test_ipk) # Menyimpan nilai hasil prediksi SVR
df_results['SVR Prediction'] = SVR_predict

# Menambahkan kolom yang berisi selisih antara IPK prediksi dan IPK asli
df_results['Selisih_IPK_SVR'] = df_results['SVR Prediction'] - df_results['IPK'] #Prediksi IPK - IPK asli
df_results.head()
```

Out[75]:	IPK	SVR Prediction	Selisih_IPK_SVR
517	3.292500	3.385216	0.092716
87	3.423255	3.406148	-0.017107
82	3.582500	3.524201	-0.058299
228	2.778571	3.021841	0.243270
349	3.516666	3.162125	-0.354541

```
In [76]: #Cek deskripsi data  
df_results.describe()
```

	IPK	SVR Prediction	Selisih_IPK_SVR
count	177.000000	177.000000	177.000000
mean	3.392731	3.414983	0.022252
std	0.325741	0.158178	0.303120
min	2.509090	3.014466	-0.553777
25%	3.175000	3.308083	-0.190443
50%	3.423255	3.406724	-0.021622
75%	3.639473	3.520363	0.197376
max	4.000000	3.825886	0.945331

## GBR (Gradient Boosting Regressor)

- Buatlah pipeline yang terdiri dari data scaling dan regressor
  - Buatlah parameter grid untuk parameter algoritme GBR
  - Buatlah GSCV yang digunakan untuk mencari model terbaik dengan metode validasi 5-fold CV
  - Tampilkan best model, nilai koefisien yang diperoleh dengan menggunakan atribut coef dan nilai intercept dari persamaan linier dengan menggunakan atribut intercept
  - Tampilkan Mean Absolute Error, Mean Squared Error, dan Root Mean Squared Error

In [81]:

```
#import library yang diperlukan untuk model GBR
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
from sklearn.metrics import mean_absolute_error, mean_squared_error #Digunakan untuk menghitung MSE dan MAE yang berguna untuk evaluasi model
```

```
#Mendefinisikan pipeline untuk GBR
pipe_GBR = Pipeline(steps=[
    ('scale', StandardScaler()),
    ('feature_selection', SelectKBest(score_func=f_regression)), #Seleksi fitur menggunakan SelectKBest dengan f_regression sebagai fungsi skor
    ('reg', GradientBoostingRegressor())
])
```

```
#Mendefinisikan parameter grid untuk GBR
param_grid_GBR = {
    'reg_n_estimators': [100, 200, 300], # Jumlah pohon dalam ensemble
    'reg_learning_rate': [0.01, 0.1, 0.2], # Mengontrol kontribusi setiap pohon
    'reg_max_depth': [3, 4, 5], # Kedalaman maksimal pohon
    'feature_selection_k': np.arange(1,20)
}
```

```
#Mengimplementasikan GridSearchCV untuk GBR
GSCV_GBR = GridSearchCV(pipe_GBR, param_grid_GBR, cv=5, scoring='neg_mean_squared_error') #'neg_mean_squared_error' digunakan untuk menghitung MSE,
#semakin kecil MSE, MAE, dan RMSE, maka semakin baik
#juga modelnya
```

```
GSCV_GBR.fit(X_train_ipk, y_train_ipk)

#Mencetak model terbaik dan parameter terbaik
print("Best model: {}".format(GSCV_GBR.best_estimator_))
print("GBR best parameters: {}".format(GSCV_GBR.best_params_))

# Mengakses feature importances dari model GBR terbaik
print("Feature Importances: {}".format(GSCV_GBR.best_estimator_.named_steps['reg'].feature_importances_))

#Membuat Prediksi
GBR_predict = GSCV_GBR.predict(X_test_ipk)

#Menghitung dan mencetak metrik evaluasi
mse_GBR = mean_squared_error(y_test_ipk, GBR_predict)
mae_GBR = mean_absolute_error(y_test_ipk, GBR_predict)

print("GBR Mean Squared Error (MSE): {}".format(mse_GBR))
print("GBR Mean Absolute Error (MAE): {}".format(mae_GBR))
print("GBR Root Mean Squared Error: {}".format(np.sqrt(mse_GBR)))
```

No Copy

No Copy

Watermarkly

```
Best model: Pipeline(steps=[('scale', StandardScaler()),  
    ('feature_selection',  
     SelectKBest(k=19,  
                 score_func=<function f_regression at 0x000001D915F3C220>)),  
    ('reg', GradientBoostingRegressor(learning_rate=0.01,  
                                      n_estimators=200))])  
GBR best parameters: {'feature_selection__k': 19, 'reg__learning_rate': 0.01, 'reg__max_depth': 3, 'reg__n_estimators': 200}  
Feature Importances: [0.05461366 0.03097433 0.12328605 0.03676886 0.0229204  0.21242032  
 0.26421593 0.04773809 0.04013069 0.01251014 0.01351125 0.06962891  
 0.0337324 0.00131602 0.00233448 0.01748713 0.00720039 0.00733749  
 0.00187345]  
GBR Mean Squared Error (MSE): 0.08847747605727885  
GBR Mean Absolute Error (MAE): 0.23827830100781755  
GBR Root Mean Squared Error: 0.2974516365012619
```

- Tambahkan hasil prediksi model GBR pada dataframe hasil

```
In [82]: #Menambahkan prediksi ke dalam data frame hasil  
df_results['GBR Prediction'] = GBR_predict  
df_results = pd.DataFrame(y_test_ipk) # Menyimpan nilai hasil prediksi SVR  
df_results['GBR Prediction'] = GBR_predict  
  
# Menambahkan kolom yang berisi selisih antara IPK prediksi dan IPK asli  
df_results['Selisih_IPK_GBR'] = df_results['GBR Prediction'] - df_results['IPK'] #Prediksi IPK - IPK asli  
df_results.head()
```

```
Out[82]:
```

	IPK	GBR Prediction	Selisih_IPK_GBR
517	3.292500	3.342523	0.050023
87	3.423255	3.478708	0.055453
82	3.582500	3.410150	-0.172350
228	2.778571	3.093700	0.315129
349	3.516666	3.205380	-0.311286

```
In [83]: #Cek deskripsi data  
df_results.describe()
```

```
Out[83]:
```

	IPK	GBR Prediction	Selisih_IPK_GBR
count	177.000000	177.000000	177.000000
mean	3.392731	3.399912	0.007180
std	0.325741	0.123737	0.298209
min	2.509090	3.033041	-0.563836
25%	3.175000	3.339929	-0.208324
50%	3.423255	3.397661	-0.030058
75%	3.639473	3.473456	0.178927
max	4.000000	3.685663	0.878987

- Membandingkan dataframe hasil dari setiap model yang sudah di latih sebelumnya

No Copy

No Copy

No Copy

No Copy

Watermarkly

```
In [84]: # Membuat dataframe untuk menampung nilai IPK asli dan prediksi dari model
df_results = pd.DataFrame({'IPK': y_test_ipk})

# Menambahkan kolom prediksi Ridge
df_results['Ridge Prediction'] = Ridge_predict
# Menambahkan kolom selisih antara prediksi Ridge dengan IPK asli
df_results['Selisih_IPK_RR'] = df_results['IPK'] - df_results['Ridge Prediction']

# Menambahkan kolom prediksi Lasso
df_results['Lasso Prediction'] = Lasso_predict
# Menambahkan kolom selisih antara prediksi Lasso dengan IPK asli
df_results['Selisih_IPK_LR'] = df_results['IPK'] - df_results['Lasso Prediction']

# Menambahkan kolom prediksi SVR
df_results['SVR Prediction'] = SVR_predict
# Menambahkan kolom selisih antara prediksi SVR dengan IPK asli
df_results['Selisih_IPK_SVR'] = df_results['IPK'] - df_results['SVR Prediction']

# Menambahkan kolom prediksi GBR
df_results['GBR Prediction'] = GBR_predict
# Menambahkan kolom selisih antara prediksi GBR dengan IPK asli
df_results['Selisih_IPK_GBR'] = df_results['IPK'] - df_results['GBR Prediction']

# Menampilkan dataframe dengan kolom IPK asli, prediksi Ridge, prediksi Lasso, dan selisih masing-masing
df_results.head()
```

```
Out[84]:
```

	IPK	Ridge Prediction	Selisih_IPK_RR	Lasso Prediction	Selisih_IPK_LR	SVR Prediction	Selisih_IPK_SVR	GBR Prediction	Selisih_IPK_GBR
517	3.292500	3.338426	-0.045926	3.344446	-0.051946	3.385216	-0.092716	3.342523	-0.050023
87	3.423255	3.442874	-0.019619	3.410330	0.012925	3.406148	0.017107	3.478708	-0.055453
82	3.582500	3.475864	0.106636	3.466212	0.116288	3.524201	0.058299	3.410150	0.172350
228	2.778571	3.038370	-0.259799	3.038451	-0.259880	3.021841	-0.243270	3.093700	-0.315129
349	3.516666	3.159569	0.357097	3.165573	0.351093	3.162125	0.354541	3.205380	0.311286

```
In [85]: #Cek deskripsi data
df_results.describe()
```

```
Out[85]:
```

	IPK	Ridge Prediction	Selisih_IPK_RR	Lasso Prediction	Selisih_IPK_LR	SVR Prediction	Selisih_IPK_SVR	GBR Prediction	Selisih_IPK_GBR
count	177.000000	177.000000	177.000000	177.000000	177.000000	177.000000	177.000000	177.000000	177.000000
mean	3.392731	3.393127	-0.000396	3.392159	0.000572	3.414983	-0.022252	3.399912	-0.007180
std	0.325741	0.153206	0.302056	0.146426	0.302298	0.158178	0.303120	0.123737	0.298209
min	2.509090	3.038370	-0.969770	3.034900	-0.934173	3.014466	-0.945331	3.033041	-0.878987
25%	3.175000	3.284496	-0.174394	3.285882	-0.167788	3.308083	-0.197376	3.339929	-0.178927
50%	3.423255	3.391923	0.043074	3.382823	0.041176	3.406724	0.021622	3.397661	0.030058
75%	3.639473	3.488273	0.201274	3.482827	0.194580	3.520363	0.190443	3.473456	0.208324
max	4.000000	3.850325	0.577734	3.814728	0.570863	3.825886	0.553777	3.685663	0.563836

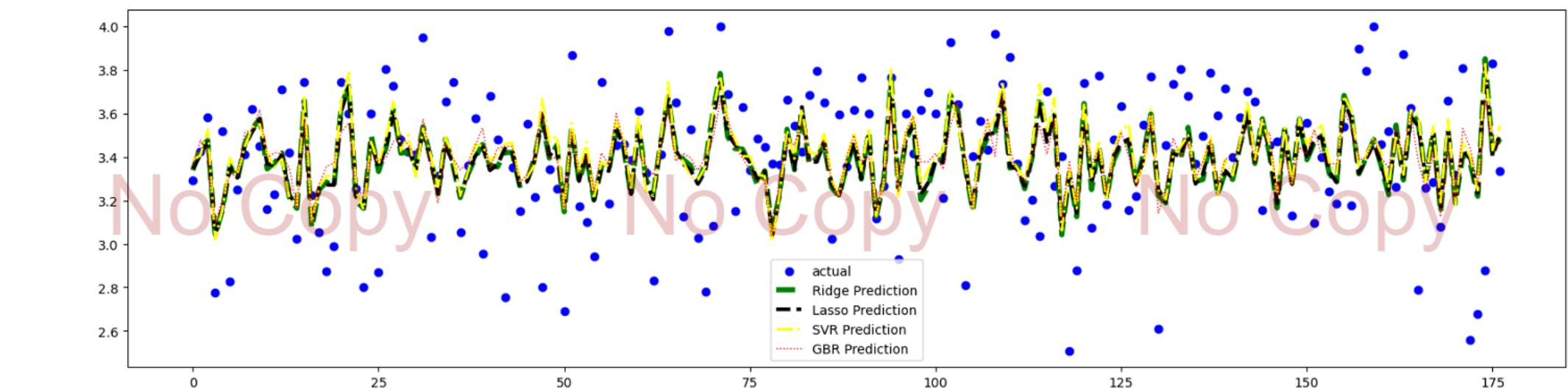
- Buatlah grafik untuk menunjukkan perbandingan antara data asli, hasil prediksi model Ridge Regression, Lasso Regression, SVR, dan GBR dari dataframe hasil

In [86]:

```
import matplotlib.pyplot as plt

#mengatur ukuran plot
plt.figure(figsize=(20,5))
#membuat rentang untuk sumbu x berdasarkan panjang data y_test_ipk
data_len = range(len(y_test_ipk))
#membuat scatter plot untuk nilai aktual (IPK) dengan warna biru
plt.scatter(data_len, df_results.IPK, label="actual", color="blue")
#membuat plot untuk prediksi ridge dengan warna hijau dan jenis garis dashed
plt.plot(data_len, df_results['Ridge Prediction'], label="Ridge Prediction", color="green", linewidth=4, linestyle="dashed")
#membuat plot untuk prediksi lasso dengan warna hitam dan jenis garis --
plt.plot(data_len, df_results['Lasso Prediction'], label="Lasso Prediction", color="black", linewidth=3, linestyle="--")
#membuat plot untuk prediksi SVR dengan warna kuning dan jenis garis _.
plt.plot(data_len, df_results['SVR Prediction'], label="SVR Prediction", color="yellow", linewidth=2, linestyle="-.")
#membuat plot untuk prediksi GBT dengan warna merah dan jenis garis :
plt.plot(data_len, df_results['GBT Prediction'], label="GBT Prediction", color="red", linewidth=1, linestyle=":")
plt.legend()
plt.show()
```

Out[86]:



- Untuk memilih model terbaik berdasarkan seberapa mirip prediksinya dengan data asli (kolom IPK sebagai target), kita dapat menggunakan beberapa metrik evaluasi regresi.

No Copy

No Copy

Watermarkly

```
In [89]: from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Hitung MAE dan RMSE serta jumlah fitur untuk Ridge
mae_ridge = mean_absolute_error(df_results['IPK'], df_results['Ridge Prediction'])
rmse_ridge = np.sqrt(mean_squared_error(df_results['IPK'], df_results['Ridge Prediction']))
ridge_feature_count = GCSV_RR.best_params_['feature_selection_k'] # Akses best_params dari GCSV_RR

# Hitung MAE dan RMSE serta jumlah fitur untuk Lasso
mae_lasso = mean_absolute_error(df_results['IPK'], df_results['Lasso Prediction'])
rmse_lasso = np.sqrt(mean_squared_error(df_results['IPK'], df_results['Lasso Prediction']))
lasso_feature_count = GCSV_Lasso.best_params_['feature_selection_k'] # Akses best_params dari GCSV_Lasso

# Hitung MAE dan RMSE serta jumlah fitur untuk SVR
mae_svr = mean_absolute_error(df_results['IPK'], df_results['SVR Prediction'])
rmse_svr = np.sqrt(mean_squared_error(df_results['IPK'], df_results['SVR Prediction']))
svr_feature_count = GCSV_SVR.best_params_['feature_selection_k'] # Akses best_params dari GCSV_SVR

# Hitung MAE dan RMSE serta jumlah fitur untuk GBR
mae_gbr = mean_absolute_error(df_results['IPK'], df_results['GBR Prediction'])
rmse_gbr = np.sqrt(mean_squared_error(df_results['IPK'], df_results['GBR Prediction']))
gbr_feature_count = GCSV_GBR.best_params_['feature_selection_k'] # Akses best_params dari GCSV_GBR

# Cetak hasil
print(F'Ridge MAE: {mae_ridge}, Ridge RMSE: {rmse_ridge}, Ridge Feature Count: {ridge_feature_count}')
print(F'Lasso MAE: {mae_lasso}, Lasso RMSE: {rmse_lasso}, Lasso Feature Count: {lasso_feature_count}')
print(F'SVR MAE: {mae_svr}, SVR RMSE: {rmse_svr}, SVR Feature Count: {svr_feature_count}')
print(F'GBR MAE: {mae_gbr}, GBR RMSE: {rmse_gbr}, GBR Feature Count: {gbr_feature_count}')
```

```
Ridge MAE: 0.23833548926056355, Ridge RMSE: 0.3012014264714568, Ridge Feature Count: 15
Lasso MAE: 0.24013325069277436, Lasso RMSE: 0.3014436727541665, Lasso Feature Count: 15
SVR MAE: 0.23851169425571525, SVR RMSE: 0.3030809558920265, SVR Feature Count: 7
GBR MAE: 0.23827830100781755, GBR RMSE: 0.2974516365012619, GBR Feature Count: 19
```

- Dump model dengan MAE dan/atau RMSE terendah, sesuaikan dengan hasil yang ada pada model anda
- Setelah melakukan pelatihan dan pengujian model, kita dapat menyimpan model menggunakan library Pickle

```
In [90]: import pickle

best_model = GCSV_SVR.best_estimator_ # Karena SVR memiliki MAE/RMSE paling rendah dan fitur yang paling sedikit,
# maka ini menunjukkan bahwa model ini memberikan prediksi yang lebih
# akurat dan efisien secara keseluruhan.

# Simpan model ke file .pkl
with open('SVR_IPK_model.pkl', 'wb') as f:
    pickle.dump(best_model, f)

print("Model terbaik berhasil disimpan ke 'SVR_IPK_model.pkl'")
```

```
Model terbaik berhasil disimpan ke 'SVR_IPK_model.pkl'
```

## Streamlit

- Buat file baru dengan ekstensi **Python (.py)** untuk membuat antarmuka Streamlit
- Buatlah code berikut ini

No Copy

No Copy

No Copy

No Copy

Watermarkly

```
In [ ]: import streamlit as st  
import pandas as pd  
import pickle  
import os
```

```
st.markdown("""  
    <style>  
        .main {  
            background-color: #1e1e1e;  
            color: #ffffff;  
        }  
        .stButton button {  
            background-color: #333333;  
            color: white;  
            border-radius: 5px;  
            padding: 10px 20px;  
            font-size: 18px;  
        }  
        .stTextInput, .stNumberInput input {  
            background-color: #333333;  
            border: 2px solid #ffffff;  
            color: #ffffff;  
        }  
        .stSidebar {  
            background-color: #000000;  
            color: #ffffff;  
        }  
        .stSidebar h3, .stSidebar p, .stSidebar label, .stSidebar .header-text {  
            color: #ffffff;  
        }  
        h1 {  
            font-family: 'Courier New', monospace;  
            color: #ffffff;  
        }  
        h3, p {  
            font-family: 'Arial', sans-serif;  
            color: #ffffff;  
        }  
        .uploadedFile {  
            font-family: 'Arial', sans-serif;  
            color: #cccccc;  
        }  
        .center {  
            display: flex;  
            justify-content: center;  
            align-items: center;  
        }  
        .center img {  
            width: 150px;  
            height: 100px;  
            object-fit: contain;  
        }  
    </style>  
    """", unsafe_allow_html=True)
```

```
st.markdown("""  
    <div class="center">  
          
    </div>  
    """", unsafe_allow_html=True)  
  
st.markdown("h1 style='text-align: center; color: #4CAF50;'>Prediksi IPK - XXXX</h1>", unsafe_allow_html=True) #XXXX = 4 digit npn terakhir  
st.markdown("<p style='text-align: center; color: #0073e6;'>Aplikasi ini berguna untuk memprediksi IPK berdasarkan nilai Matematika,  
Bahasa Inggris, dan Bahasa Indonesia</p>""", unsafe_allow_html=True)
```



```

# Sidebar untuk input file
st.sidebar.markdown("<h3 class='header-text'>Upload File dan Input Nilai</h3>", unsafe_allow_html=True)
uploaded_file = st.sidebar.file_uploader("Upload file dataset_regresi_IPK.csv", type=["csv"])

if uploaded_file is not None:
    # Read CSV file
    input_data = pd.read_csv(uploaded_file)
    st.write("<h3 style='text-align: center; color: #0073e6;'>Data yang diupload:</h3>", unsafe_allow_html=True)
    st.dataframe(input_data)

model_directory = r'D:\SURYA\UAJY\Semester 5\Asdos Machine Learning\Pemegang Modul\Modul Supervised Learning\New folder\New folder'
model_path = os.path.join(model_directory, r'SVR_IPK_model.pkl')

if os.path.exists(model_path):
    with open(model_path, 'rb') as f:
        loaded_model = pickle.load(f)

scaler = loaded_model[0]
feature_selector = loaded_model[1]
SVR_model = loaded_model[2]

# Sidebar input untuk nilai
st.sidebar.subheader("Masukkan Nilai")
mtk1 = st.sidebar.number_input("Nilai Matematika Semester 1.1", 60.0, 100.0)
mtk2 = st.sidebar.number_input("Nilai Matematika Semester 1.2", 60.0, 100.0)
mtk3 = st.sidebar.number_input("Nilai Matematika Semester 2.1", 68.0, 100.0)
mtk4 = st.sidebar.number_input("Nilai Matematika Semester 2.2", 66.0, 100.0)
ing1 = st.sidebar.number_input("Nilai Bahasa Inggris Semester 1.1", 64.0, 98.0)
ing2 = st.sidebar.number_input("Nilai Bahasa Inggris Semester 1.2", 62.0, 99.0)
ing3 = st.sidebar.number_input("Nilai Bahasa Inggris Semester 2.1", 68.0, 99.0)
ing4 = st.sidebar.number_input("Nilai Bahasa Inggris Semester 2.2", 68.0, 99.0)
ind1 = st.sidebar.number_input("Nilai Bahasa Indonesia Semester 1.1", 67.0, 98.0)
ind2 = st.sidebar.number_input("Nilai Bahasa Indonesia Semester 1.2", 70.0, 99.0)
ind3 = st.sidebar.number_input("Nilai Bahasa Indonesia Semester 2.1", 65.0, 99.0)
ind4 = st.sidebar.number_input("Nilai Bahasa Indonesia Semester 2.2", 70.0, 100.0)

# KKM
st.sidebar.subheader("Masukkan KKM Nilai")
kkm_ind1 = st.sidebar.number_input("KKM Nilai Bahasa Indonesia Semester 1.1", 0.0, 100.0)
kkm_ind2 = st.sidebar.number_input("KKM Nilai Bahasa Indonesia Semester 1.2", 0.0, 100.0)
kkm_ind3 = st.sidebar.number_input("KKM Nilai Bahasa Indonesia Semester 2.1", 0.0, 100.0)
kkm_ind4 = st.sidebar.number_input("KKM Nilai Bahasa Indonesia Semester 2.2", 0.0, 100.0)
kkm_ing1 = st.sidebar.number_input("KKM Nilai Bahasa Inggris Semester 1.1", 0.0, 100.0)
kkm_ing2 = st.sidebar.number_input("KKM Nilai Bahasa Inggris Semester 1.2", 0.0, 100.0)
kkm_ing3 = st.sidebar.number_input("KKM Nilai Bahasa Inggris Semester 2.1", 0.0, 100.0)
kkm_ing4 = st.sidebar.number_input("KKM Nilai Bahasa Inggris Semester 2.2", 0.0, 100.0)
kkm_mtk1 = st.sidebar.number_input("KKM Nilai Matematika Semester 1.1", 0.0, 100.0)
kkm_mtk2 = st.sidebar.number_input("KKM Nilai Matematika Semester 1.2", 0.0, 100.0)
kkm_mtk3 = st.sidebar.number_input("KKM Nilai Matematika Semester 2.1", 0.0, 100.0)
kkm_mtk4 = st.sidebar.number_input("KKM Nilai Matematika Semester 2.2", 0.0, 100.0)

input_data = [mtk1, mtk2, mtk3, mtk4, ing1, ing2, ing3, ing4, ind1, ind2, ind3, ind4,
            kkm_mtk1, kkm_mtk2, kkm_mtk3, kkm_mtk4, kkm_ing1, kkm_ing2, kkm_ing3, kkm_ing4,
            kkm_ind1, kkm_ind2, kkm_ind3, kkm_ind4]

input_data_scaled = scaler.transform([input_data])
input_data_selected = feature_selector.transform(input_data_scaled)

if st.sidebar.button("Prediksi!"):
    SVR_model_predict = SVR_model.predict(input_data_selected)
    st.markdown(f"<h3 style='text-align: center; color: #4CAF50;'>Prediksi IPK adalah: {SVR_model_predict[0]:.2f}</h3>", unsafe_allow_html=True)
else:
    st.error("Model tidak ditemukan, silakan cek file model di direktori.")

```

- Buka terminal dengan cara **CTRL + Shift + `**, kemudian ketik **conda activate base**
  - Untuk menjalankan proyek streamlit, ketikan **streamlit run (path file)**
  - Contoh : **streamlit run "D:\Kuliah\Semester 5\streamlit.py"**
  - Ketika dijalankan, secara otomatis akan terbuka tab baru di browser
- 
- Setelah streamlit berhasil dibuat. Upload file ke github dan lakukan deploy secara online
  - Tutorial deploy online terdapat di situs kuliah

Format Penamaan:

- Tugas\_Modul3\_X\_YYY\_ZZZZZ.ipynb
- Tugas\_Modul3\_X\_YYY\_ZZZZZ.py
- RR/Lasso/SVR/GBR/regresi\_IPK\_model.pkl (sesuai dengan model yang terbaik)
- Tugas\_Modul3\_X\_YYY\_ZZZZZ.txt (berisi link streamlit yang sudah di deploy secara online)

a. X untuk kelas

b. Y untuk Nama panggilan Praktikan

c. Z untuk 5 digit NPM

Contoh : Tugas\_Modul3\_A\_MARIA\_11969.ipynb

Semua file di kumpul dalam 1 folder kemudian di zip

Penamaan Folder Zip Tugas3\_X\_YYYYY.zip

a. X untuk kelas

b. Y untuk 5 digit NPM

Contoh : Tugas3\_A\_11969.zip