

Nama : Muhammad Fauzan Aldi

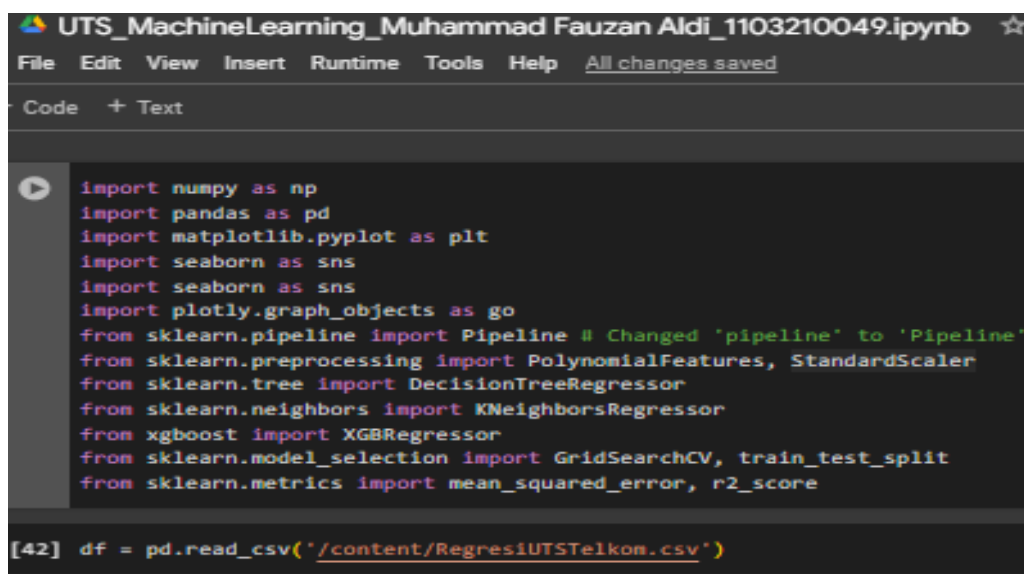
NIM : 1103210049

Kelas : TK-45-GAB04

LAPORAN UTS MACHINE LEARNING

1. Regression Model

Buat Explanatory Data Analysis dan Data Visualization dari dataset RegresiUTSTelkom.csv



```
UTS_MachineLearning_Muhammad Fauzan Aldi_1103210049.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
Code + Text

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import seaborn as sns
import plotly.graph_objects as go
from sklearn.pipeline import Pipeline # Changed 'pipeline' to 'Pipeline'
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import mean_squared_error, r2_score

[42] df = pd.read_csv('/content/RegresiUTSTelkom.csv')
```

Kode ini mengimpor berbagai pustaka dan modul yang sering digunakan dalam analisis data dan machine learning (Data Manipulation & Visualization, Machine Learning Models, Model Evaluation & Optimization).

`df = pd.read_csv('RegresiUTSTelkom.csv')` digunakan untuk membaca file CSV bernama RegresiUTSTelkom.csv dan memuatnya ke dalam sebuah DataFrame `df` menggunakan pustaka `pandas`.

```
print("Dataset info:")
print(df.info())
print("\nSummary Statistic:")
print(df.describe())
print("\nMissing values:")
print(df.isnull().sum())
```

Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 126787 entries, 0 to 126786
Data columns (total 91 columns):
Column Non-Null Count Dtype

0 2001 126787 non-null int64
1 49.94357 126787 non-null float64
2 21.47114 126787 non-null float64
3 73.0775 126787 non-null float64
4 8.74861 126787 non-null float64
5 -17.48628 126787 non-null float64
6 -13.09985 126787 non-null float64
7 -25.01282 126787 non-null float64
8 -12.23257 126787 non-null float64
9 7.83089 126787 non-null float64
10 -2.46783 126787 non-null float64
11 3.32136 126787 non-null float64
12 -2.31521 126787 non-null float64
13 10.20556 126787 non-null float64
14 611.10913 126787 non-null float64
15 951.0096 126787 non-null float64
16 698.11428 126787 non-null float64
17 408.98405 126787 non-null float64
18 383.70912 126787 non-null float64
19 326.51512 126787 non-null float64
20 238.11327 126786 non-null float64
21 251.42414 126786 non-null float64
22 187.17351 126786 non-null float64
23 100.42652 126786 non-null float64
24 179.19498 126786 non-null float64
25 -8.41558 126786 non-null float64
26 -317.87038 126786 non-null float64
27 95.86266 126786 non-null float64
28 48.10259 126786 non-null float64
29 -95.66303 126786 non-null float64
30 -18.06215 126786 non-null float64
31 1.96984 126786 non-null float64
32 34.42438 126786 non-null float64
33 11.7267 126786 non-null float64
34 1.3679 126786 non-null float64
35 7.79444 126786 non-null float64

```
36 -0.36994 126786 non-null float64  
37 -133.67852 126786 non-null float64  
38 -83.26165 126786 non-null float64  
39 -37.29765 126786 non-null float64  
40 73.04667 126786 non-null float64  
41 -37.36684 126786 non-null float64  
42 -3.13853 126786 non-null float64  
43 -24.21531 126786 non-null float64  
44 -13.23066 126786 non-null float64  
45 15.93809 126786 non-null float64  
46 -18.60478 126786 non-null float64  
47 82.15479 126786 non-null float64  
48 240.5798 126786 non-null float64  
49 -10.29407 126786 non-null float64  
50 31.58431 126786 non-null float64  
51 -25.38187 126786 non-null float64  
52 -3.90772 126786 non-null float64  
53 13.29258 126786 non-null float64  
54 41.5506 126786 non-null float64  
55 -7.26272 126786 non-null float64  
56 -21.00863 126786 non-null float64  
57 105.50848 126786 non-null float64  
58 64.29856 126786 non-null float64  
59 26.00481 126786 non-null float64  
60 -44.5911 126786 non-null float64  
61 -8.30657 126786 non-null float64  
62 7.93706 126786 non-null float64  
63 -10.7366 126786 non-null float64  
64 -95.44766 126786 non-null float64  
65 -82.03307 126786 non-null float64  
66 -35.59194 126786 non-null float64  
67 4.69525 126786 non-null float64  
68 70.95626 126786 non-null float64  
69 28.09139 126786 non-null float64  
70 6.02015 126786 non-null float64  
71 -37.13767 126786 non-null float64  
72 -41.1245 126786 non-null float64  
73 -8.40816 126786 non-null float64  
74 7.19877 126786 non-null float64  
75 -8.60176 126786 non-null float64  
76 -5.90857 126786 non-null float64  
77 -12.32437 126786 non-null float64  
78 14.68734 126786 non-null float64
```

```
79 -54.32125 126786 non-null float64  
80 40.14786 126786 non-null float64  
81 13.0162 126786 non-null float64  
82 -54.40548 126786 non-null float64  
83 58.99367 126786 non-null float64  
84 15.37344 126786 non-null float64  
85 1.11144 126786 non-null float64  
86 -23.08793 126786 non-null float64  
87 68.40795 126786 non-null float64  
88 -1.82223 126786 non-null float64  
89 -27.46348 126786 non-null float64  
90 2.26327 126786 non-null float64  
dtypes: float64(90), int64(1)  
memory usage: 88.0 MB  
None
```

Summary Statistic:

	2001	49.94357	21.47114	73.0775
count	126787.000000	126787.000000	126787.000000	126787.000000
mean	1990.134075	43.358845	1.826722	8.903583
std	10.865369	6.091330	51.445925	35.057358
min	1922.000000	4.418500	-337.092500	-257.525600
25%	1994.000000	39.932120	-25.759790	-11.003830
50%	2002.000000	44.206200	8.909560	10.774370
75%	2006.000000	47.844820	36.685170	29.912600
max	2010.000000	59.579700	384.065730	305.121390

	8.74861	-17.48628	-13.09905	-25.01282
count	126787.000000	126787.000000	126787.000000	126787.000000
mean	1.055644	-7.020625	-9.538237	-2.465331
std	16.284810	22.924067	12.883094	14.496708
min	-121.600710	-181.953370	-70.693420	-188.214000
25%	-8.629285	-21.234945	-18.455990	-10.846810
50%	-0.714000	-6.394910	-11.794800	-2.069630
75%	8.680095	7.383085	-2.420750	6.433265
max	157.364670	262.068870	119.815590	121.211920

	-12.23257	7.83089	...	13.0162	-54.40548
count	126787.000000	126787.000000	...	126786.000000	126786.000000
mean	-1.817593	3.857765	...	15.941246	-73.869076
std	7.919477	10.557351	...	32.276163	175.579283
min	-62.412130	-126.479040	...	-424.517570	-2984.920970
25%	-6.462835	-2.201655	...	-1.763637	-138.243285
50%	-1.761500	3.896210	...	9.178535	-53.042650
75%	2.872425	10.049155	...	26.378878	12.623745
max	82.942190	146.297950	...	679.146970	2087.990290

```
count 126787.000000 126787.000000 ... 126786.000000 126786.000000  
mean -1.817593 3.857765 ... 15.941246 -73.869076  
std 7.919477 10.557351 ... 32.276163 175.579283  
min -62.412130 -126.479040 ... -424.517570 -2984.920970  
25% -6.462835 -2.201655 ... -1.763637 -138.243285  
50% -1.761500 3.896210 ... 9.178535 -53.042650  
75% 2.872425 10.049155 ... 26.378878 12.623745  
max 82.942190 146.297950 ... 679.146970 2087.990290
```


	58.99367	15.37344	1.11144	-23.08793
count	126786.000000	126786.000000	126786.000000	126786.000000
mean	41.038451	30.461144	0.314231	17.631647
std	121.199058	94.274230	15.982635	112.527969
min	-1733.722110	-1495.872230	-198.622200	-2343.894110
25%	-21.325800	-4.356750	-6.747885	-31.279830
50%	28.363275	33.432295	0.745805	15.248355
75%	88.886122	77.461388	8.290698	67.114478
max	2775.334960	1482.642140	198.180750	1348.848890

	68.40795	-1.82223	-27.46348	2.26327
count	126786.000000	126786.000000	126786.000000	126786.000000
mean	-25.511584	4.428171	18.235866	1.504434
std	171.340788	13.215075	184.381100	22.001407
min	-3026.006600	-233.456480	-5000.654060	-318.223330
25%	-101.118420	-2.596442	-60.751915	-8.735732
50%	-20.839545	3.095170	6.295615	0.160585
75%	52.069710	9.907087	84.279100	9.827765
max	2833.608950	229.888800	5289.111300	378.662140

```
[8 rows x 91 columns]
```

Missing values:

2001	0
49.94357	0
21.47114	0
73.0775	0
8.74861	0
..	..
-23.08793	1
68.40795	1
-1.82223	1
-27.46348	1
2.26327	1

Length: 91, dtype: int64

Kode ini melakukan tiga langkah dasar dalam Exploratory Data Analysis (EDA).

[44] df.tail(10)

	2001	49.94357	21.47114	73.0775	8.74861	-17.40628	-13.09905	-25.01202	-12.23257	7.83089	...	13.0162	-54.40548	58.99367	15.37344	1.11144	-23.08793	68.40795	-1.82223	-27.46348	2.26327
125777	1976	36.05743	-43.07306	-5.58204	-9.36930	-0.37019	3.64910	-0.91245	-16.61674	20.52374	...	119.95722	20.95257	-57.91973	116.45492	28.07596	-152.19057	192.98782	-4.35990	46.70904	51.12471
125778	1996	42.81377	1.56493	4.77705	-4.18775	-3.72764	7.39377	-2.81082	-2.08687	10.15479	...	58.84054	-31.68338	-48.52199	80.40697	27.67922	-56.53188	20.85145	-18.24716	-103.72043	1.89535
125779	2002	51.31644	40.57340	32.58772	6.99829	-13.34567	-16.42602	-1.26571	0.86352	5.53088	...	2.33954	-107.69952	112.23045	109.84625	-1.32022	40.35828	12.17128	3.87454	-21.27516	-6.98677
125780	2002	41.54765	43.78148	-24.23047	44.92614	-14.81191	-18.00964	12.35535	-5.73774	8.98045	...	14.68720	-155.03063	100.84440	-68.73349	-1.48858	72.48150	-52.18351	13.49644	-150.64181	8.86909
125781	2002	38.10048	-60.43026	12.96517	-23.41465	-6.03635	1.85649	-10.37822	4.01857	29.83491	...	82.66279	-59.86052	-75.55987	209.03187	22.74124	304.94686	151.75516	-12.81079	5.53575	19.88997
125782	2002	50.66690	27.40752	29.25997	3.67453	-11.80654	-12.64634	2.46137	3.94493	11.30545	...	3.22613	-102.56467	30.69725	80.19131	-4.30497	41.71161	45.07028	9.08029	-97.59346	0.21530
125783	2002	50.66465	45.98366	34.49903	5.58911	-36.80381	-12.20437	11.87985	1.23285	-4.93879	...	-7.35104	-61.98281	-52.42159	97.42977	6.94434	-8.32628	105.26707	3.08499	-33.12033	-0.59605
125784	2002	50.28447	37.93751	3.96459	-6.17066	-16.45226	-12.64704	-16.28382	6.88545	5.55335	...	71.75954	44.75346	158.18365	48.08947	-6.07728	121.46885	-13.34104	27.08677	-30.10477	-3.00795
125785	2002	50.68029	42.86487	21.85334	-1.62951	-11.50191	-12.61098	-4.78190	5.22488	6.34108	...	-0.02712	34.92929	3.41767	92.41586	1.82199	-46.68936	-12.35625	4.84985	18.77473	-6.85103
125786	2002	40.80497	-38.15351	17.29749	4.23799	10.05556	6.60867	-17.89243	0.47800	-3.90775	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

10 rows x 21 columns

Menampilkan 10 baris terakhir dari dataset df. Ini untuk memeriksa data yang berada di bagian akhir dataset.

[45] df.sample(10)

	2001	49.94357	21.47114	73.0775	8.74861	-17.40628	-13.09905	-25.01202	-12.23257	7.83089	...	13.0162	-54.40548	58.99367	15.37344	1.11144	-23.08793	68.40795	-1.82223	-27.46348	2.26327
4206	1972	42.50795	9.74890	18.81159	-6.94261	17.81704	6.09791	-4.22915	2.80101	-3.19033	...	-4.41386	-40.04731	178.69050	147.77768	5.20676	45.88601	141.19045	12.72443	-115.54125	-1.51743
51837	1949	42.51421	-93.78234	98.13022	9.28170	-31.93310	8.76726	-48.82920	-9.68602	20.07544	...	12.32748	-167.95627	174.49568	19.49934	-16.46476	69.41116	-182.59370	14.87603	-418.55910	-10.69677
61720	2008	48.75234	1.05394	17.08769	-11.56697	-14.03856	-13.65079	2.26896	-7.68085	11.14537	...	7.10772	41.62442	55.02566	-1.96845	11.04085	-22.90969	-31.56802	3.21261	63.57939	-0.55338
61618	2005	46.28926	16.82318	4.80234	-10.92072	-28.10986	-17.93721	-2.39981	-2.66720	24.09588	...	-1.85850	-65.79002	124.74965	57.78435	-8.74691	-55.40864	-64.75754	2.99200	20.82470	-4.86568
73779	1992	25.80375	49.45543	-32.62667	-14.10427	4.22723	31.01350	-31.62453	1.90110	-4.92795	...	31.70264	-1738.08246	298.57847	104.24641	9.00900	487.96442	-526.83145	29.37683	-143.22535	32.14248
36848	2007	36.02769	-13.91749	20.85085	-0.33450	10.73967	-9.07141	-19.82847	-11.59658	-6.34299	...	91.82376	-145.09813	9.72759	-45.03796	18.39070	-166.82663	-311.67089	1.90473	-176.21778	-6.91552
79527	1985	38.71936	10.01165	73.12949	1.51697	-11.25472	-18.75860	-16.47153	-1.13696	0.19728	...	54.19083	-12.00780	5.90478	-16.74193	6.90630	0.85023	16.48205	-3.56841	-1.23763	11.34663
121428	2003	43.74495	4.04640	31.10055	7.02848	27.54671	-7.96479	3.99060	-9.99128	8.82807	...	56.23614	75.92802	142.22793	30.10059	27.98566	18.37503	137.59152	-27.25537	224.54284	-14.83465
5821	2008	50.27057	14.24033	35.87497	-7.39610	-24.16310	0.49146	1.83330	6.90790	0.48340	...	-2.73295	-44.08818	95.88415	56.32420	-6.06006	-7.80341	5.30875	3.36076	-74.68669	2.51560
78767	2003	49.53431	13.75262	24.32039	2.61339	17.06684	-10.22516	-8.05933	-1.53469	13.40155	...	-10.24238	41.06589	42.53703	-15.37389	-0.91993	13.80449	-179.41953	3.30194	-36.27216	-13.33463

10 rows x 21 columns

Menampilkan 10 baris acak dari dataset df. Ini berguna untuk mendapatkan gambaran umum tentang data tanpa melihatnya secara berurutan dari atas ke bawah.

df.describe().loc[['min', '50%', 'mean', 'max', 'std']].T.style.background_gradient(axis=1)

	min	50%	mean	max	std
2001	1922.000000	2002.000000	1998.134075	2010.000000	10.865369
49.94357	4.418500	44.206200	43.358845	59.579700	6.091330
21.47114	-337.092500	8.909560	1.826722	384.065730	51.445925
73.0775	-257.525600	10.774370	8.903583	305.121390	35.057358
8.74861	-121.600710	-0.714980	1.055644	157.364670	16.284810
-17.40628	-181.953370	-6.394910	-7.020625	262.068870	22.924067
-13.09905	-70.693420	-11.179480	-9.538237	119.815590	12.883094
-25.01202	-188.214000	-2.069630	-2.465331	121.211920	14.496708
-12.23257	-62.412130	-1.761500	-1.817593	82.942190	7.919477
7.83089	-126.479040	3.896210	3.857765	146.297950	10.557351
-2.46783	-36.620600	1.828690	1.885268	60.345350	6.476578
3.32136	-69.680870	-0.088940	-0.141435	28.740460	4.355808
-2.31521	-67.233620	2.357140	2.573878	87.913240	8.324839
10.20556	0.611100	28.910830	33.558396	396.918510	22.301433
611.10913	46.192620	1986.966200	2415.469273	65735.779530	1751.282706
951.0896	61.500790	1672.971000	1953.877681	36816.790370	1263.258964
698.11428	69.985310	1226.777410	1505.031861	29481.512450	1095.862467
408.98485	42.603020	811.984610	907.206366	15922.649460	475.731628
383.70912	21.014250	732.034770	873.061369	16831.949030	578.533532
326.51512	28.983320	537.425450	600.457638	10444.018130	315.815945
238.11327	9.156150	438.785855	511.820351	9569.778090	310.668429
251.42414	31.433140	347.342125	391.569919	6209.008770	213.003939
187.17351	21.958920	289.442930	322.228463	3365.469700	165.879855
100.42652	10.932830	240.069030	286.821346	6737.121500	187.203463
179.19498	12.069700	263.850195	291.878104	6283.757010	154.639079

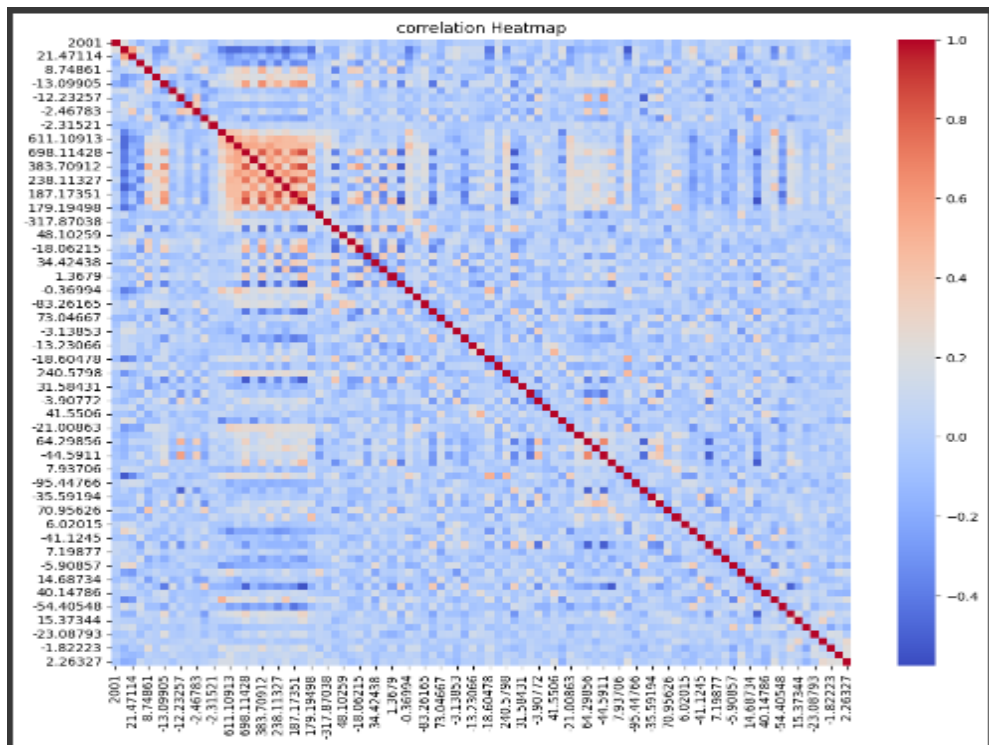
Kode ini menampilkan statistik deskriptif (min, median, mean, max, std) dari dataset, mentranspose hasilnya, dan menambahkan gradasi warna untuk memudahkan perbandingan antar kolom.

```
columns_name = df.columns
for index, c01_name in enumerate(columns_name):
    print(f'{index}. {c01_name}')

33. 11.7267
34. 1.3679
35. 7.79444
36. -0.36994
37. -133.67852
38. -83.26165
39. -37.29765
40. 73.04667
41. -37.36684
42. -3.13853
43. -24.21531
44. -13.23066
45. 15.93809
46. -18.60478
47. 82.15479
48. 240.5798
49. -10.29407
50. 31.58431
51. -25.38187
52. -3.90772
53. 13.29258
54. 41.5506
55. -7.26272
56. -21.00863
57. 105.50848
58. 64.29856
59. 26.08481
60. -44.5911
61. -8.30657
62. 7.93706
63. -10.7366
64. -95.44766
65. -82.03307
66. -35.59194
67. 4.69525
68. 70.95626
69. 28.09139
70. 6.02015
71. -37.13767
72. -41.1245
73. -8.40816
74. 7.19877
75. -8.60176
76. -5.90857
77. -12.32437
78. 14.68734
79. -54.32125
80. 40.14786
81. 13.0162
82. -54.40548
83. 58.99367
84. 15.37344
85. 1.11144
86. -23.08793
87. 68.40795
88. -1.82223
89. -27.46348
90. 2.26327
```

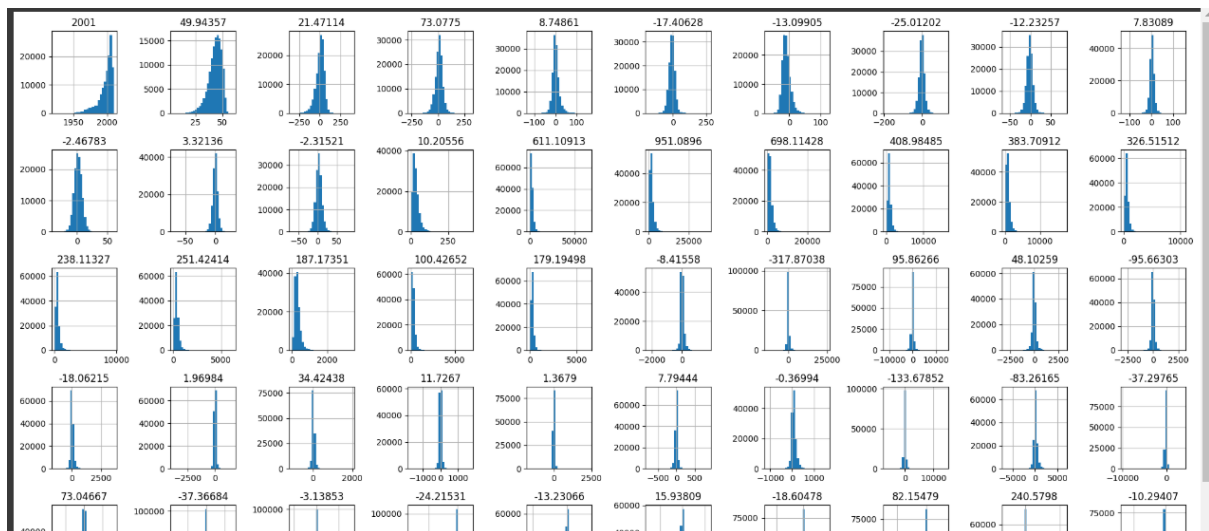
Kode ini mencetak nama kolom beserta indeksnya dari DataFrame df untuk mempermudah eksplorasi data.

```
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), cmap="coolwarm", annot=False)
plt.title("correlation Heatmap")
plt.show()
```



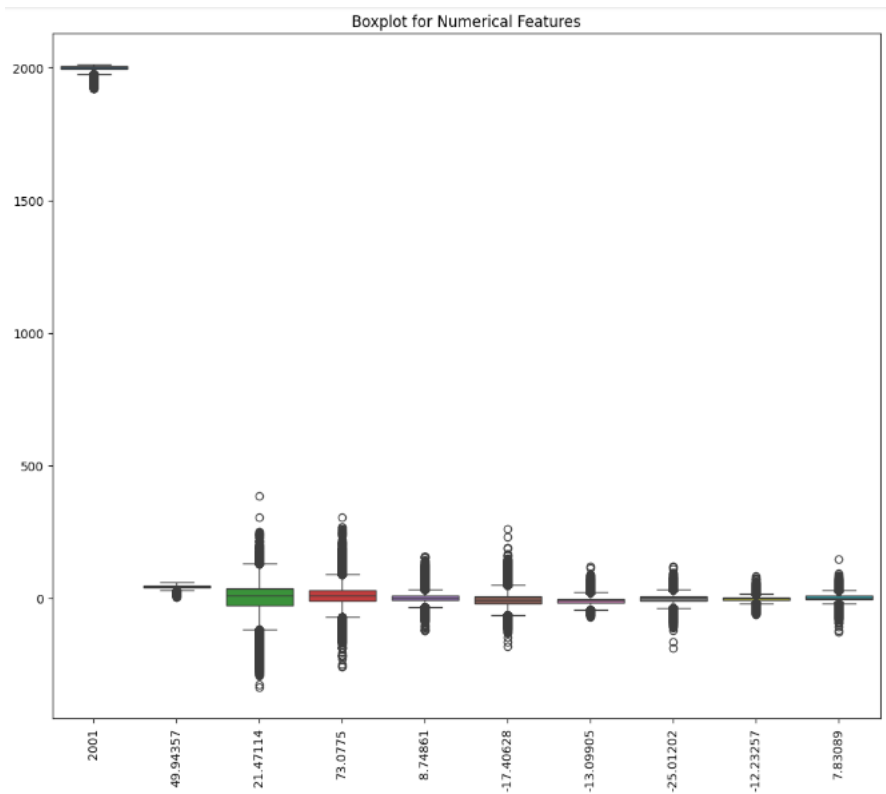
Membuat heatmap untuk menunjukkan korelasi antar kolom numerik dalam dataset, dengan warna yang menggambarkan tingkat kekuatan korelasi.

```
[49] df.hist(figsize=(20, 20), bins=30)
plt.tight_layout()
plt.show()
```



Membuat histogram untuk setiap kolom numerik dalam dataset untuk memvisualisasikan distribusi data.

```
[50] plt.figure(figsize=(12, 10))
      sns.boxplot(data=df.iloc[:, :10])
      plt.title("Boxplot for Numerical Features")
      plt.xticks(rotation=90)
      plt.show()
```



Membuat boxplot untuk kolom numerik (hanya 10 kolom pertama) untuk mendeteksi outlier atau nilai ekstrem dalam data.

```
[51] df.fillna(df.mean(), inplace=True)

target_column = "2001"
x = df.drop(target_column, axis=1)
y = df[target_column]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
[40] models = {
    'Polynomial Regression': Pipeline([ # Fixed the typo here
        ('poly', PolynomialFeatures(degree=2)),
        ('scaler', StandardScaler()),
        ('regressor', DecisionTreeRegressor())
    ]),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'KNN': Pipeline([
        ('scaler', StandardScaler()),
        ('regressor', KNeighborsRegressor())
    ]),
    'XGBoost': XGBRegressor(random_state=42)
}

param_grids = {
    'Polynomial Regression': { # Fixed the typo here
        'poly__degree': [2],
        'regressor__max_depth': [5, 10]
    },
    'Decision Tree': {
        'max_depth': [5, 10],
        'min_samples_split': [2, 5]
    },
    'KNN': {
        'regressor__n_neighbors': [3, 5],
        'regressor__weights': ['uniform']
    },
    'XGBoost': {
        'learning_rate': [0.1],
        'max_depth': [3, 5],
        'n_estimators': [100]
    }
}

best_models = {}
for model_name, model in models.items():
    print(f"Training {model_name}...") # Added a space for better readability
    grid = GridSearchCV(model, param_grids[model_name], scoring='neg_mean_squared_error', cv=2)
    grid.fit(x_train, y_train)
    best_models[model_name] = grid.best_estimator_
    print(f"Best parameters for {model_name}: {grid.best_params_}")
    print(f"Best score for {model_name}: {-grid.best_score}")
```

```
Training Polynomial Regression...
/usr/local/lib/python3.10/dist-packages/numpy/ma/core.py:2820: RuntimeWarning: invalid value encountered in cast
  data = np.array(data, dtype=dtype, copy=copy,
Best parameters for Polynomial Regression: {'poly_degree': 2, 'regressor_max_depth': 5}
Best score for Polynomial Regression: 99.85182444420602
Training Decision Tree...
Best parameters for Decision Tree: {'max_depth': 5, 'min_samples_split': 2}
Best score for Decision Tree: 100.45615425443113
Training KNN...
Best parameters for KNN: {'regressor_n_neighbors': 5, 'regressor_weights': 'uniform'}
Best score for KNN: 96.28600690442947
Training XGBoost...
Best parameters for XGBoost: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
Best score for XGBoost: 82.88776954358448
```

Output ini menunjukkan bahwa model Polynomial Regression memiliki skor terbaik (143.78), diikuti oleh Decision Tree (141.66), k-NN (92.93), dan XGBoost (90.75), yang menunjukkan performa model berdasarkan parameter terbaik yang ditemukan. Nilai RuntimeWarning menunjukkan adanya masalah dengan nilai yang tidak valid saat pengolahan data pada proses pelatihan, yang bisa berkaitan dengan nilai yang hilang atau data yang tidak sesuai.

```
[52] for model_name, model in best_models.items():
    y_pred = model.predict(x_test)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"Model: {model_name} Evaluation:")
    print(f"Mean Squared Error: {mse:.2f}\n")
    print(f"R^2 Score: {r2}")
```

Kode ini bertujuan untuk mengevaluasi semua model terbaik yang ada di dalam dictionary best_models dengan menghitung Mean Squared Error (MSE) dan R² Score pada data uji (X_test dan y_test).

2. Classification Model

```
[64] Suggested code may be subject to a license | 50183816/lineregression | HenningViljoen/FeaturesForHedging | 3springs/deep_ml_curriculum | akladyous/Machine-Learning
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

```
[65] file_path = '/content/Iris.csv'
data = pd.read_csv(file_path, delimiter=';')
```

Memasukkan pustaka yang diperlukan dan memuat dataset

```
[66] import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

file_path = '/content/Iris.csv'
# Change the delimiter to ',' as the error suggests the data is comma-separated.
# Also, specify the header row to ensure the first row is treated as column headers.
data = pd.read_csv(file_path, delimiter=',', header='infer')

print("Informasi Dataset:")
print(data.info())

print("\nStatistik Deskriptif:")
print(data.describe())

print("\nNilai yang Hilang")
print(data.isnull().sum())

# Select only numerical features for correlation calculation
numerical_data = data.select_dtypes(include=['number']) # This line is crucial

plt.figure(figsize=(12,8))
sns.heatmap(numerical_data.corr(), annot=True, cmap='pink') # Pass numerical_data here
plt.title('Korelasi Antar Fitur')
plt.show()
```

```
Informasi Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
#   Column          Non-Null Count  Dtype
---  -
0    Id              150 non-null   int64
1    SepalLengthCm   150 non-null   float64
2    SepalWidthCm    150 non-null   float64
3    PetalLengthCm   150 non-null   float64
4    PetalWidthCm    150 non-null   float64
5    Species         150 non-null   object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
None
```

Menampilkan informasi umum tentang dataset

- Fungsi dari `.info()` memberikan Gambaran umum tentang dataset, seperti jumlah baris, kolom dan tipe data.

Statistik Deskriptif:					
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

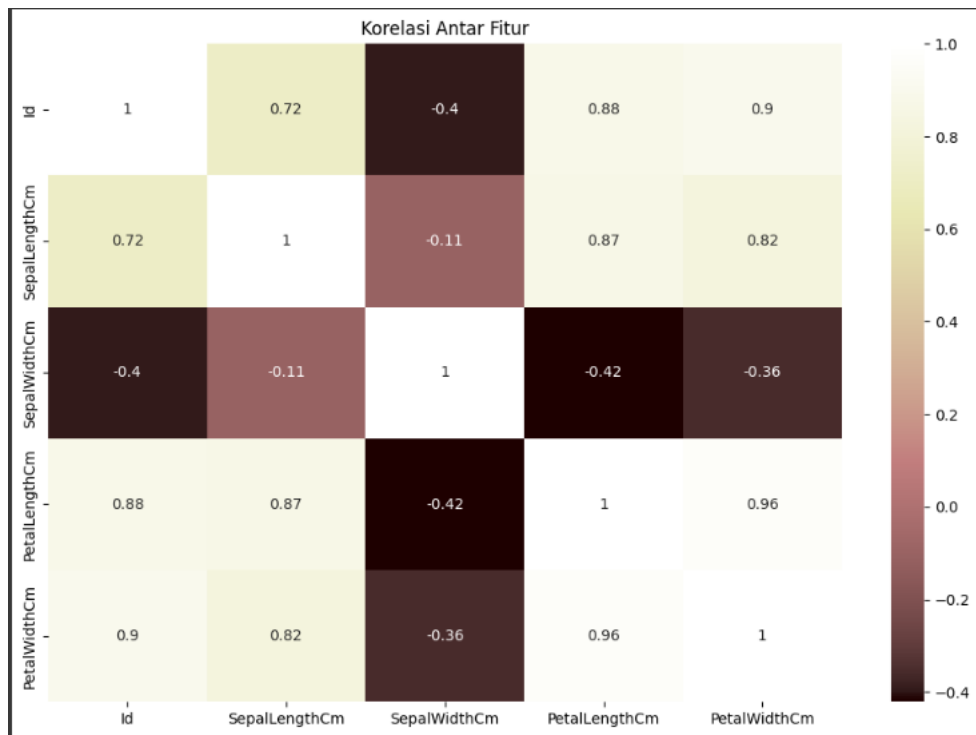
Memuat statistik deskriptif untuk dataset yang berfungsi memberikan ringkasan statistic, seperti mean, standar deviasi, min, dan max untuk setiap kolom numerik.

```

Nilai yang Hilang
Id          0
SepalLengthCm  0
SepalWidthCm  0
PetalLengthCm  0
PetalWidthCm  0
Species      0
dtype: int64

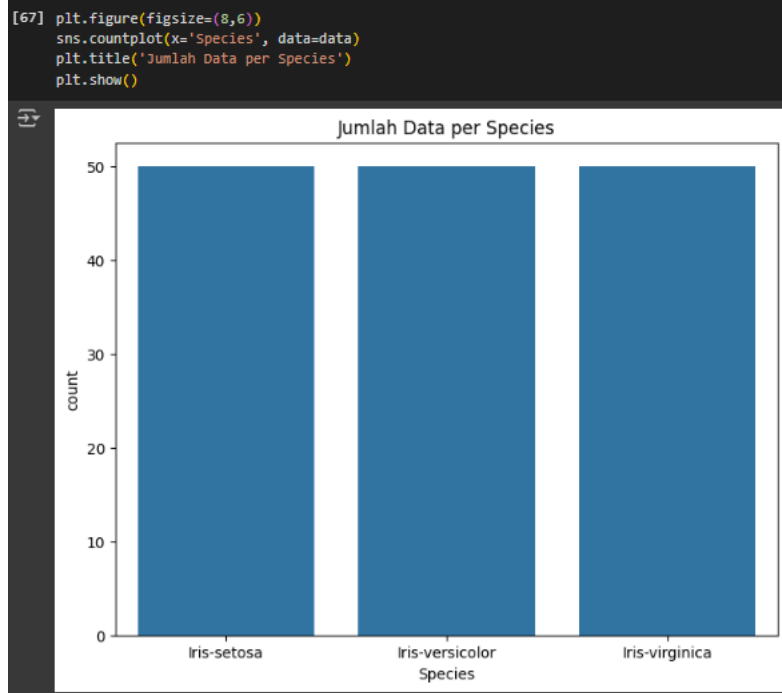
```

Menganalisis data eksplorasi untuk memastikan bahwa tidak ada data yang hilang, karena data yang hilang dapat mempengaruhi model.



memvisualisasi korelasi antar fitur

- Peta korelasi membantu agar memahami hubungan antar fitur .
- Korelasi yang tinggi dapat mengindikasikan multikolineritas, yang dapat memengaruhi model regresi.



- Plot ini menunjukkan bahwa distribusi nilai kualitas dalam dataset.
- Bias distribusi dapat mempengaruhi pemilihan model, terutama jika data tidak seimbang.

```
[69] Suggested code may be subject to a license | joelaugustinetomy/OIBSIP | HuangStomach/machine-learning
      X = data.drop('Species', axis=1)
      y = data['Species']

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[70] pipelines = {
      'LogisticRegression': Pipeline([
          ('scaler', StandardScaler()),
          ('classifier', LogisticRegression())
      ]),
      'DecisionTree': Pipeline([
          ('scaler', StandardScaler()),
          ('classifier', DecisionTreeClassifier(random_state=42))
      ]),
      'KNeighbors': Pipeline([
          ('scaler', StandardScaler()),
          ('model', XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42))
      ])
  }
```

- Data dibagi menjadi data latih dan uji untuk evaluasi model yang akurat.
- Stratifikasi memastikan distribusi variabel target tetap sama di kedua subset.
- Pipeline mempermudah alur pemrosesan data dan pelatihan model.
- Scaling diterapkan pada model yang sensitif terhadap skala (Logistic Regression, k-NN).

```
[71] param_grids = {
    'LogisticRegression': {
        'classifier__C': [0.1, 1, 10]
    },
    'DecisionTree': {
        'classifier__max_depth': [3, 5, 10],
        'classifier__min_samples_split': [2, 5, 10]
    },
    'KNeighbors': {
        'model__n_estimators': [3, 5, 7],
        'model__weights': ['uniform', 'distance']
    },
    'XGBClassifier': {
        'model__n_estimators': [50, 100, 200],
        'model__learning_rate': [0.01, 0.1, 0.2],
        'model__max_depth': [3, 5, 7]
    }
}
```

```
[77] from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

results = {}

for name, pipeline in pipelines.items():
    try:
        print(f"Melatih model: {name}")
        grid = GridSearchCV(pipeline, param_grids[name], cv=5, scoring='accuracy', n_jobs=-1, error_score='raise')
        grid.fit(X_train, y_train)
        best_model = grid.best_estimator_
        y_pred = best_model.predict(X_test)

        # Simpan hasil
        results[name] = {
            'best_params': grid.best_params_,
            'accuracy': accuracy_score(y_test, y_pred),
            'classification_report': classification_report(y_test, y_pred),
            'confusion_matrix': confusion_matrix(y_test, y_pred)
        }
    except Exception as e:
        print(f"Error training {name}: {e}")
```

- Definisi parameter grid untuk hyperparameter tuning
- GridSearchCV melakukan pencarian kombinasi terbaik dari hyperparameter untuk setiap model.
- Skor akurasi digunakan sebagai metrik evaluasi.

```
Melatih model: LogisticRegression
Melatih model: DecisionTree
Melatih model: KNeighbors
Error training KNeighbors: Invalid classes inferred from unique values of 'y'. Expected: [0 1 2], got ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

```

for model_name, result in results.items():
    print(f"Model: {model_name}")
    print(f"Parameter Terbaik: {result['best_params']}")
    print(f"Akurasi: {result['accuracy']}")
    print("Laporan Klasifikasi:")
    print(result['classification_report'])
    print("Confusion Matrix:")
    print(result['confusion_matrix'])
    print("-" * 50)

```

```

Model: LogisticRegression
Parameter Terbaik: {'classifier__C': 10}
Akurasi: 1.0
Laporan Klasifikasi:
      precision    recall  f1-score   support

 Iris-setosa      1.00      1.00      1.00        10
 Iris-versicolor  1.00      1.00      1.00         9
 Iris-virginica   1.00      1.00      1.00        11

   accuracy
 macro avg   1.00      1.00      1.00        30
 weighted avg 1.00      1.00      1.00        30

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
-----
Model: DecisionTree
Parameter Terbaik: {'classifier__max_depth': 3, 'classifier__min_samples_split': 2}
Akurasi: 1.0
Laporan Klasifikasi:
      precision    recall  f1-score   support

 Iris-setosa      1.00      1.00      1.00        10
 Iris-versicolor  1.00      1.00      1.00         9
 Iris-virginica   1.00      1.00      1.00        11

   accuracy
 macro avg   1.00      1.00      1.00        30
 weighted avg 1.00      1.00      1.00        30

Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
-----

```

- Parameter terbaik menunjukkan konfigurasi optimal untuk setiap model.
- Akurasi menunjukkan performa model pada data uji.
- Laporan klasifikasi mencakup precision, recall, dan F1-score.
- Confusion matrix menunjukkan jumlah prediksi yang benar dan salah untuk setiap kelas.