

Nama: Rizki Aprilia Rahman

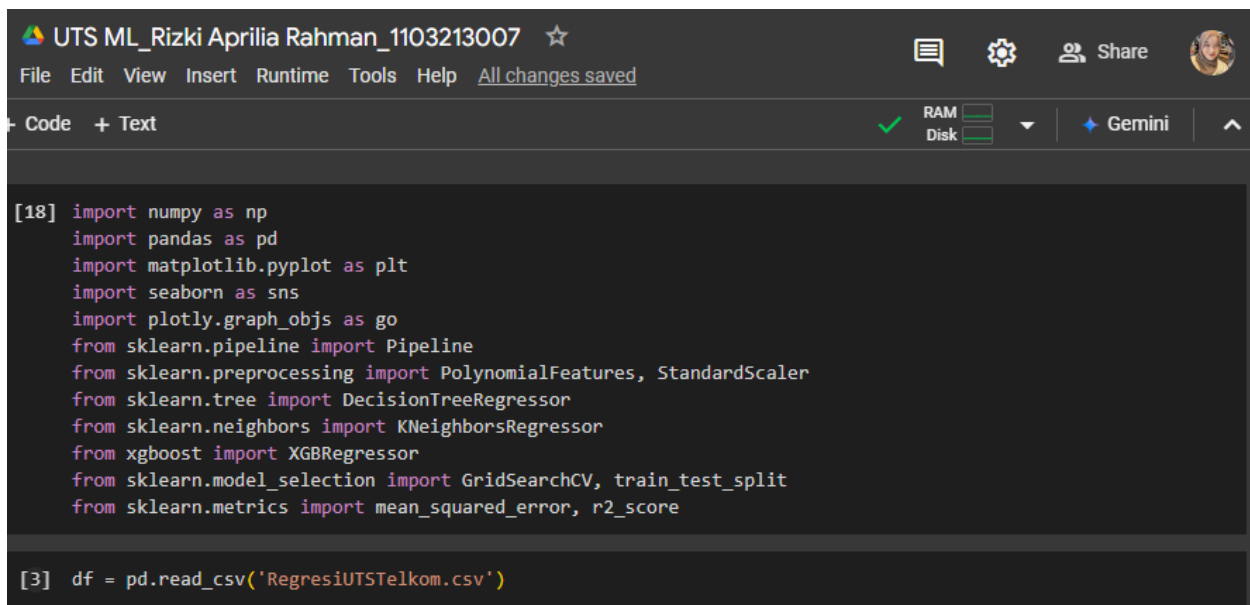
NIM: 1103213007

Kelas: TK-45-GAB04

## LAPORAN UTS PEMBELAJARAN MESIN

### 1. Regression Model

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



The screenshot shows a Jupyter Notebook window titled "UTS ML\_Rizki Aprilia Rahman\_1103213007". The interface includes a top bar with menu items (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar indicating "All changes saved". Below the menu is a toolbar with icons for Code, Text, RAM, Disk, and a Gemini button. The main area contains two code cells. The first cell, labeled [18], imports various libraries: numpy, pandas, matplotlib, seaborn, plotly, and several machine learning models from sklearn (Pipeline, PolynomialFeatures, StandardScaler, DecisionTreeRegressor, KNeighborsRegressor, XGBRegressor, GridSearchCV, train\_test\_split, mean\_squared\_error, r2\_score). The second cell, labeled [3], uses pandas to read the 'RegresiUTSTelkom.csv' file into a DataFrame named 'df'.

```
[18] import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
from sklearn.pipeline import 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

[3] df = pd.read_csv('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).

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

```
# EDA: Basic Information
print("Dataset Info:")
print(df.info())
print("\nSummary Statistics:")
print(df.describe())
print("\nMissing Values:")
print(df.isnull().sum())
```

Dataset Info:  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 2436 entries, 0 to 2435  
Data columns (total 91 columns):  
# Column Non-Null Count Dtype  
---  
0 2001 2436 non-null int64  
1 49.94357 2436 non-null float64  
2 21.47114 2436 non-null float64  
3 73.0775 2436 non-null float64  
4 8.74861 2436 non-null float64  
5 -17.40628 2436 non-null float64  
6 -13.09905 2436 non-null float64  
7 -25.01202 2436 non-null float64  
8 -12.23257 2436 non-null float64  
9 7.83089 2436 non-null float64  
10 -2.46783 2436 non-null float64  
11 3.32136 2436 non-null float64  
12 -2.31521 2436 non-null float64  
13 10.20556 2436 non-null float64  
14 611.10913 2436 non-null float64  
15 951.0896 2436 non-null float64  
16 698.11428 2436 non-null float64  
17 408.98485 2436 non-null float64  
18 383.70912 2436 non-null float64  
19 326.51512 2436 non-null float64  
20 238.11327 2436 non-null float64  
21 251.42414 2436 non-null float64  
22 187.17351 2436 non-null float64  
23 100.42652 2436 non-null float64  
24 179.19498 2436 non-null float64  
25 -8.41558 2436 non-null float64  
26 -317.87038 2436 non-null float64  
27 95.86266 2436 non-null float64  
28 48.10259 2436 non-null float64  
29 -95.66303 2436 non-null float64  
...

59 26.08481 2435 non-null float64  
60 -44.5911 2435 non-null float64  
61 -8.30657 2435 non-null float64  
62 7.93706 2435 non-null float64  
63 -10.7366 2435 non-null float64  
64 -95.44766 2435 non-null float64  
65 -82.03307 2435 non-null float64  
66 -35.59194 2435 non-null float64  
67 4.69525 2435 non-null float64  
68 70.95626 2435 non-null float64  
69 28.09139 2435 non-null float64  
70 6.02015 2435 non-null float64  
71 -37.13767 2435 non-null float64  
72 -41.1245 2435 non-null float64  
73 -8.40816 2435 non-null float64  
74 7.19877 2435 non-null float64  
75 -8.60176 2435 non-null float64  
76 -5.90857 2435 non-null float64  
77 -12.32437 2435 non-null float64  
78 14.68734 2435 non-null float64  
79 -54.32125 2435 non-null float64  
80 40.14786 2435 non-null float64  
81 13.0162 2435 non-null float64  
82 -54.40548 2435 non-null float64  
83 58.99367 2435 non-null float64  
84 15.37344 2435 non-null float64  
85 1.11144 2435 non-null float64  
86 -23.08793 2435 non-null float64  
87 68.40795 2435 non-null float64  
88 -1.82223 2435 non-null float64  
89 -27.46348 2435 non-null float64  
90 2.26327 2435 non-null float64  
...

dtypes: float64(90), int64(1)  
memory usage: 1.7 MB  
None

```
Summary Statistics:
```

	2001	49.94357	21.47114	73.0775	8.74861
count	2436.000000	2436.000000	2436.000000	2436.000000	2436.000000
mean	1999.137931	43.616425	1.215844	8.831503	4.309915
std	10.954557	5.981817	52.090356	36.417153	17.257849
min	1930.000000	18.982540	-265.265710	-161.801660	-70.041280
25%	1995.000000	40.331073	-22.932253	-10.790658	-6.429440
50%	2003.000000	44.577775	10.974230	11.320960	1.690390
75%	2007.000000	48.057617	35.613795	30.846438	12.164590
max	2010.000000	54.009400	171.163460	233.167660	112.688390

	-17.40628	-13.09905	-25.01202	-12.23257	7.83089
count	2436.000000	2436.000000	2436.000000	2436.000000	2436.000000
mean	-7.969649	-9.013343	-0.406793	-1.771605	4.098386
std	23.278170	14.194519	14.184564	8.484667	9.968570
min	-91.857230	-55.861450	-96.656410	-46.321820	-70.018700
25%	-22.737053	-18.001055	-8.584655	-6.609080	-1.332240
50%	-6.538260	-10.808670	0.085210	-1.485910	4.054370
75%	6.657138	-2.103755	8.536025	3.247140	9.035215
max	262.068870	65.624630	111.529610	47.619120	75.038510

	13.0162	-54.40548	58.99367	15.37344	1.11144
count	2435.000000	2435.000000	2435.000000	2435.000000	2435.000000
mean	15.707827	-68.681454	39.611093	44.456161	-0.972982
std	31.538213	166.969813	132.803370	110.583572	16.243777
min	-255.007250	-1650.756890	-700.599910	-562.645010	-130.674300
25%	-1.664480	-132.696800	-26.459075	-1.653535	-8.173785
50%	9.257990	-51.553860	26.605890	36.544440	-0.100810
75%	25.251870	14.768950	86.952330	81.167640	7.283950
max	1125.880950	814.336930	203.649480	2559.572110	274.984190

	-23.08793	68.40795	-1.82223	-27.46348	2.26327
count	2435.000000	2435.000000	2435.000000	2435.000000	2435.000000
mean	14.757489	-24.363751	4.878729	22.736549	1.745666
std	122.894157	169.479488	15.400011	187.663083	22.058939
min	-745.366140	-1699.235270	-64.538940	-1511.396930	-177.403100
25%	-30.738220	-93.556510	-2.929185	-60.562415	-8.185490
50%	15.922250	-19.947380	3.103260	6.340230	0.103800
75%	67.799680	50.852760	10.525120	80.001355	9.978590
max	1125.880950	814.336930	203.649480	2559.572110	274.984190

[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).

0s [21] `df.tail(10)`

	2001	49.94357	21.47114	73.0775	8.74861	-17.40628	-13.09905	-25.01202	-12.23257	7.83089	..
2426	1993	39.52388	-4.10973	-21.47340	-16.05970	16.80924	-16.310490	-23.100950	-16.800320	14.362440	
2427	1993	35.90717	-39.13095	-11.91888	-6.59343	1.16663	-5.654020	-9.386440	-8.003210	12.426160	
2428	1993	36.77905	-23.31036	-30.50373	-23.73960	-10.62307	-6.487810	-10.981870	-0.004660	3.880440	
2429	1993	30.88855	-23.48024	-9.78307	-11.66286	21.21963	-11.952040	-17.600480	-4.760830	-3.123000	
2430	1993	36.79051	0.20232	-14.15370	4.48215	8.91114	-0.404230	-10.722460	-2.515700	17.886540	
2431	2003	37.25759	-70.15967	-50.37725	-25.21705	15.28944	-10.293850	-19.334310	-16.827440	6.824450	
2432	2003	40.02856	-30.40858	-39.26240	-18.82759	8.70231	-7.331310	-7.908330	-15.833170	7.774150	
2433	2003	38.75749	-49.74751	-55.05697	-13.33387	16.33493	3.407610	-1.419180	-9.165190	1.461940	
2434	2003	39.71746	-32.07159	-40.99851	-19.86263	2.75615	-15.493600	-8.792740	-17.743050	4.062870	
2435	1996	32.87646	-15.91552	-89.04646	-14.42151	14.15240	-9.013343	-0.406793	-1.771605	4.098386	

10 rows x 91 columns

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

0s `df.sample(10)`

	2001	49.94357	21.47114	73.0775	8.74861	-17.40628	-13.09905	-25.01202	-12.23257	7.83089	...
330	1998	36.53393	-25.38981	-13.21255	-20.43358	-14.28570	-26.18190	0.28586	-5.35317	9.39508	..
1045	2002	50.89676	3.81508	54.95638	-1.34448	24.46904	-19.77313	-9.87140	-1.65469	2.60130	..
1888	1976	41.91901	-15.39226	20.38791	4.65013	-7.60598	-11.30604	34.71235	5.05834	9.79537	..
1500	1994	40.14436	20.21266	60.92652	-5.10605	17.72906	-4.00209	-27.26257	-0.81212	4.35778	..
1394	1995	43.47451	2.36132	-29.52697	-20.28861	14.50479	-16.56017	-16.47265	-9.21591	1.26739	..
1138	2010	44.74138	51.30258	-36.85797	9.10249	-7.65854	-25.90110	0.33339	7.00571	2.26803	..
1325	1999	48.75211	23.55673	34.76566	28.73099	18.80590	-5.16373	3.24864	-6.38860	1.51853	..
2226	2009	49.73356	-17.98949	49.33562	16.62149	-5.53746	-8.89618	11.27385	-0.99269	-6.70004	..
853	1997	42.63125	18.74280	71.68144	5.97359	38.02159	4.62209	-58.44378	-12.06097	-22.67690	..
1589	2003	41.38912	-40.31909	17.51815	6.16085	-24.65497	-4.85650	12.29182	-2.86757	2.88521	..

10 rows x 91 columns

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

+ Code + Text

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

	min	50%	mean	max	std
2001	1930.000000	2003.000000	1999.137931	2010.000000	10.954557
49.94357	18.982540	44.577775	43.616425	54.009400	5.981817
21.47114	-265.265710	10.974230	1.215844	171.163460	52.090356
73.0775	-161.801660	11.320960	8.831503	233.167660	36.417153
8.74861	-70.041280	1.690390	4.309915	112.688390	17.257849
-17.40628	-91.857230	-6.538260	-7.969649	262.068870	23.278170
-13.09905	-55.861450	-10.896375	-9.013343	65.624630	14.191604
-25.01202	-96.656410	0.079690	-0.406793	111.529610	14.181651
-12.23257	-46.321820	-1.491050	-1.771605	47.619120	8.482925
7.83089	-70.918790	4.058620	4.098386	75.038510	9.966523
-2.46783	-28.097500	2.439025	2.393530	29.478050	6.419890
3.32136	-23.923970	0.030225	0.021204	18.721930	4.489520
-2.31521	-40.818190	1.530815	1.974645	41.010240	8.318165
10.20556	1.692850	28.407085	32.609628	396.918510	21.813743
611.10913	149.835960	1939.301830	2376.931631	24682.600740	1776.369421
951.0896	116.495510	1657.279325	1990.933330	11894.549520	1338.162510
698.11428	122.594710	1274.583120	1557.749317	17997.793530	1137.360813
408.98485	136.194060	783.405240	906.516468	7599.205590	521.939648
383.70912	54.347640	775.748975	945.582189	4919.823690	603.517972
326.51512	115.072480	537.157980	608.172875	4795.046950	346.272002
238.11327	81.410550	454.074215	511.974108	3822.654770	286.878704
251.42414	73.302060	328.687555	374.995003	3277.015680	228.550838

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

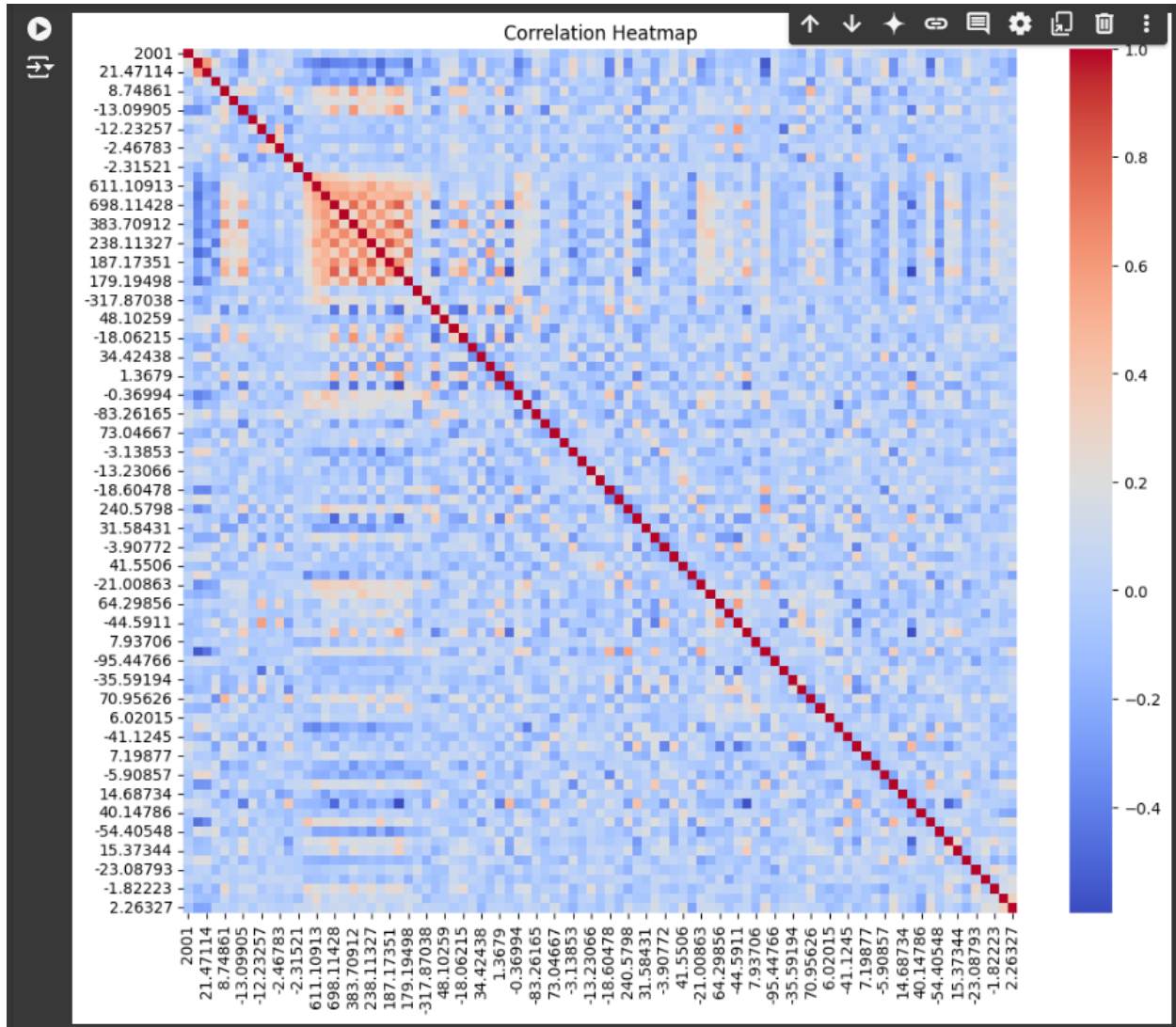
```
[24] columns_name = df.columns
      for index, c01_name in enumerate(columns_name):
          print(f'{index}. {c01_name}')
```

```
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.

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

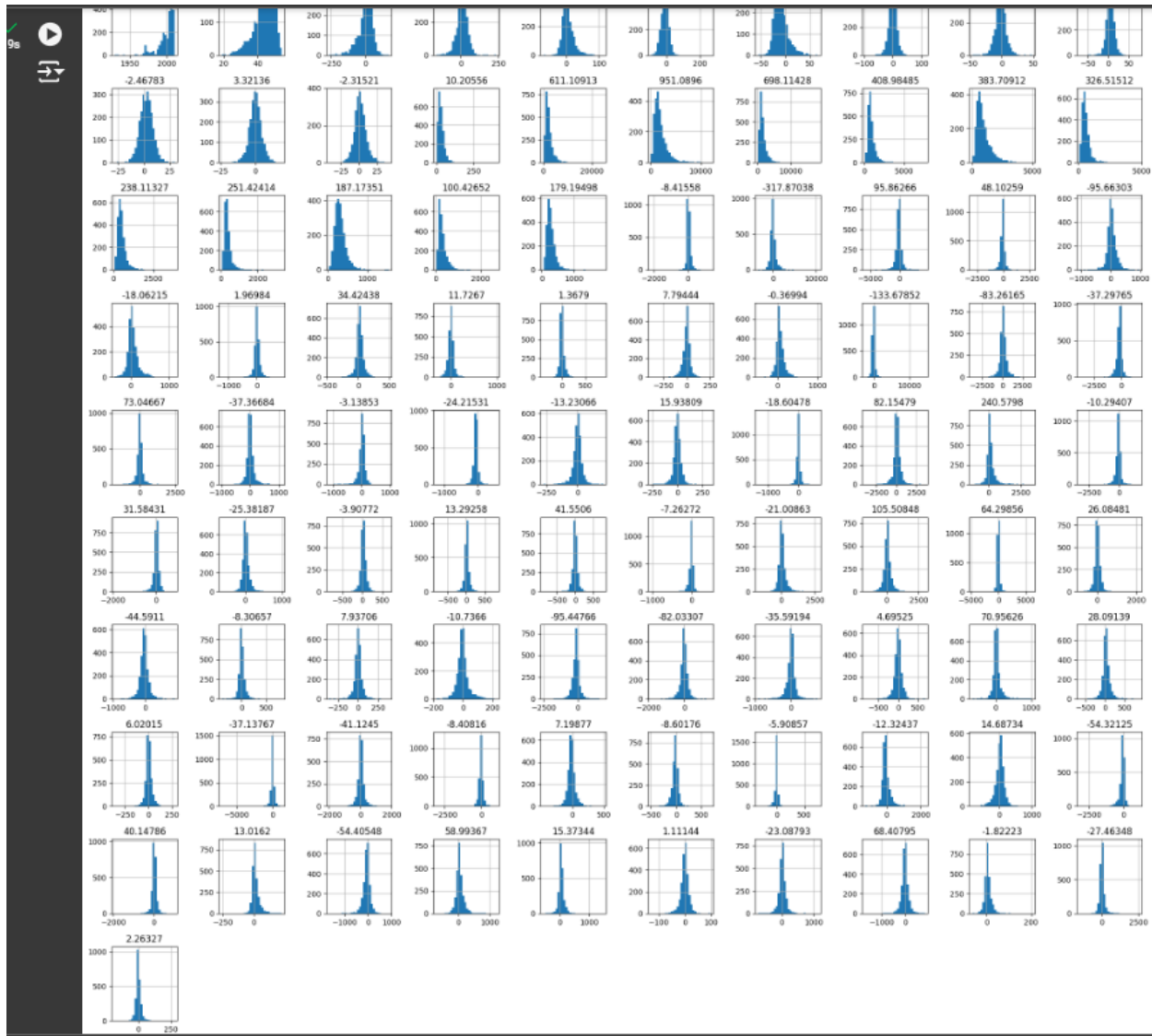


Membuat heatmap untuk menunjukkan korelasi antar kolom numerik dalam dataset, dengan warna yang menggambarkan tingkat kekuatan korelasi.

```

✓ 19s # Distribution of Numerical Columns
df.hist(figsize=(20, 20), bins=30)
plt.tight_layout()
plt.show()

```

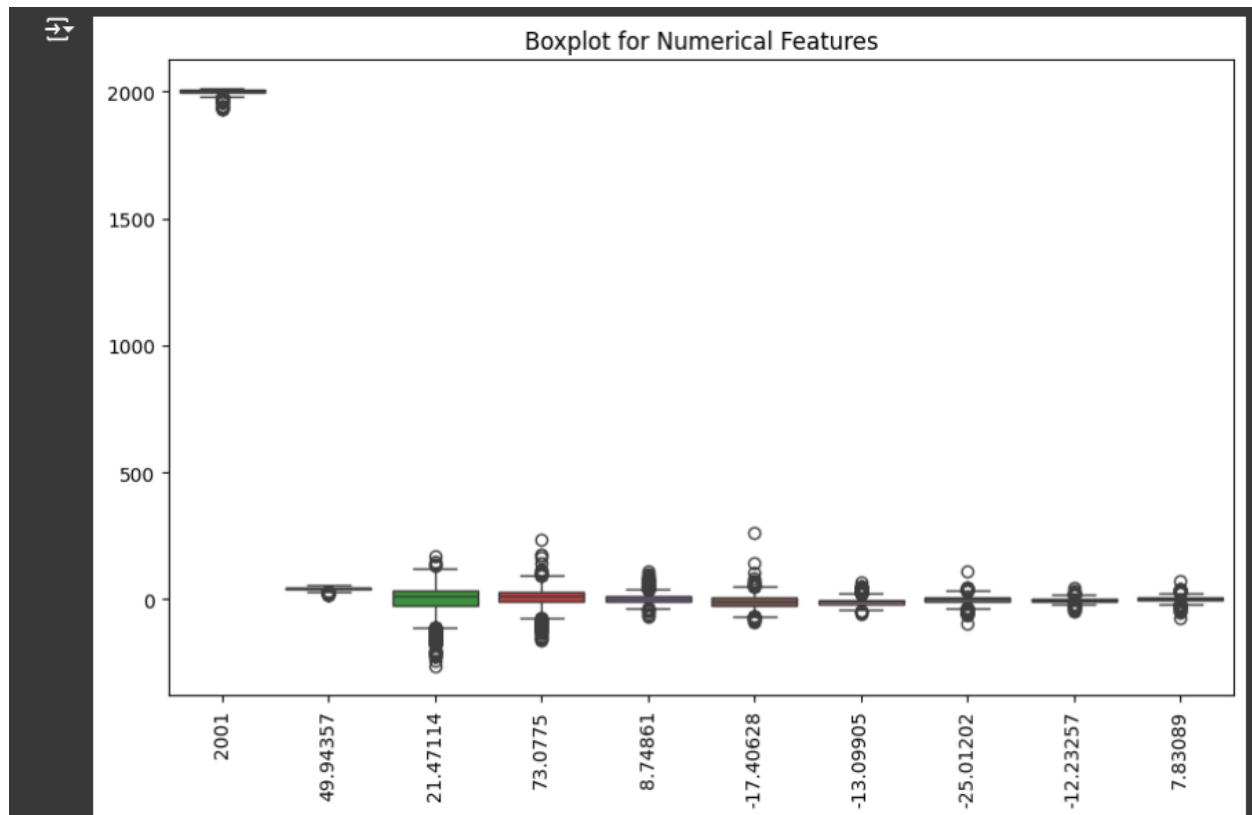


Membuat histogram untuk setiap kolom numerik dalam dataset untuk memvisualisasikan distribusi data.

```

# Boxplot for Outlier Detection
plt.figure(figsize=(10, 6))
sns.boxplot(data=df.iloc[:, :10]) # Plot only the first 10 columns for clarity
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.



```

[9] # Feature Engineering and Preprocessing

# Handle Missing Values (Impute with Mean)
df.fillna(df.mean(), inplace=True)

# Feature Selection (Optional): Select relevant features based on domain knowledge or correlation
target_column = '2001' # Assuming column '2001' is the target variable
X = df.drop(target_column, axis=1)
y = df[target_column]

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[20] # Machine Learning Pipeline

# Define pipelines for each model
models = {
    'Polynomial Regression': Pipeline([
        ('poly', PolynomialFeatures(degree=2)), # Polynomial transformation
        ('scaler', StandardScaler()), # Feature scaling
        ('regressor', DecisionTreeRegressor()) # Regression model
    ]),
    'Decision Tree': DecisionTreeRegressor(random_state=42), # Standalone model
    'k-NN': Pipeline([
        ('scaler', StandardScaler()), # Feature scaling
        ('regressor', KNeighborsRegressor()) # k-NN regression
    ]),
    'XGBoost': XGBRegressor(random_state=42) # Standalone XGBoost
}

# Hyperparameter Grids
param_grids = {
    'Polynomial Regression': {
        'poly_degree': [2], # Hanya gunakan satu nilai
        'regressor__max_depth': [5, 10] # Kurangi jumlah opsi
    },
    'Decision Tree': {
        'max_depth': [5, 10], # Kurangi jumlah opsi
        'min_samples_split': [2, 5] # Kurangi jumlah opsi
    },
    'k-NN': {
        'regressor__n_neighbors': [3, 5], # Kurangi jumlah opsi
        'regressor__weights': ['uniform'] # Fokus pada satu jenis
    },
    'XGBoost': {
        'learning_rate': [0.1], # Fokus pada satu nilai
        'max_depth': [3, 5], # Kurangi jumlah opsi
        'n_estimators': [100] # Kurangi jumlah opsi
    }
}

# Hyperparameter Tuning and Training
best_models = {}
for model_name, model in models.items():
    print(f"Training {model_name}...")
    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_}")

```

```

},
'k-NN': {
    'regressor__n_neighbors': [3, 5], # Kurangi jumlah opsi
    'regressor__weights': ['uniform'] # Fokus pada satu jenis
},
'XGBoost': {
    'learning_rate': [0.1], # Fokus pada satu nilai
    'max_depth': [3, 5], # Kurangi jumlah opsi
    'n_estimators': [100] # Kurangi jumlah opsi
}

# Hyperparameter Tuning and Training
best_models = {}
for model_name, model in models.items():
    print(f"Training {model_name}...")
    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 c
_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: 143.7815750508846
Training Decision Tree...
Best Parameters for Decision Tree: {'max_depth': 5, 'min_samples_split': 2}
Best Score for Decision Tree: 141.655895149254
Training k-NN...
Best Parameters for k-NN: {'regressor__n_neighbors': 5, 'regressor__weights': 'uniform'}
Best Score for k-NN: 92.93299794661186
Training XGBoost...
Best Parameters for XGBoost: {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100}
Best Score for XGBoost: 90.75290936607968

```

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.

```
✓ ▶ # Model Evaluation

# Evaluate all models on the test set
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_name} Evaluation:")
    print(f"    Mean Squared Error: {mse:.2f}")
    print(f"    R^2 Score: {r2:.2f}\n")
```

Kode ini bertujuan untuk mengevaluasi semua model terbaik yang ada di dalam dictionary `best_models` dengan menghitung Mean Squared Error (MSE) dan  $R^2$  Score pada data uji (`X_test` dan `y_test`).

## 2. Classification Model

```
[ ] # Import pustaka yang diperlukan
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

▶ # Langkah 1: Memuat dataset
# Dataset winequality-white.csv dimuat menggunakan delimiter ';'
file_path = 'winequality-white.csv'
data = pd.read_csv(file_path, delimiter=';')
```


Memasukkan pustaka yang diperlukan dan memuat dataset

```

# Tampilkan informasi umum tentang dataset
print("Informasi Dataset:")
print(data.info())

# Penjelasan:
# - Fungsi .info() memberikan gambaran umum tentang dataset, seperti jumlah baris, kolom, dan tipe data.
# - Informasi ini penting untuk memahami struktur data sebelum analisis lebih lanjut.

```

 Informasi Dataset:  
 <class 'pandas.core.frame.DataFrame'>  
 RangeIndex: 4898 entries, 0 to 4897  
 Data columns (total 12 columns):  

#	Column	Non-Null Count	Dtype
0	fixed acidity	4898 non-null	float64
1	volatile acidity	4898 non-null	float64
2	citric acid	4898 non-null	float64
3	residual sugar	4898 non-null	float64
4	chlorides	4898 non-null	float64
5	free sulfur dioxide	4898 non-null	float64
6	total sulfur dioxide	4898 non-null	float64
7	density	4898 non-null	float64
8	pH	4898 non-null	float64
9	sulphates	4898 non-null	float64
10	alcohol	4898 non-null	float64
11	quality	4898 non-null	int64

 dtypes: float64(11), int64(1)  
 memory usage: 459.3 KB  
 None

Menampilkan informasi umum tentang dataset.

- Fungsi .info() memberikan gambaran umum tentang dataset, seperti jumlah baris, kolom, dan tipe data.
- Informasi ini penting untuk memahami struktur data sebelum analisis lebih lanjut.

```
[ ] # Statistik deskriptif untuk dataset
print("\nStatistik Deskriptif:")
print(data.describe())

# Penjelasan:
# - Fungsi .describe() memberikan ringkasan statistik, seperti mean, standar deviasi, min, dan max untuk setiap kolom numerik.
```



Statistik Deskriptif:

	fixed acidity	volatile acidity	citric acid	residual sugar	\
count	4898.000000	4898.000000	4898.000000	4898.000000	
mean	6.854788	0.278241	0.334192	6.391415	
std	0.843868	0.100795	0.121020	5.072058	
min	3.800000	0.080000	0.000000	0.600000	
25%	6.300000	0.210000	0.270000	1.700000	
50%	6.800000	0.260000	0.320000	5.200000	
75%	7.300000	0.320000	0.390000	9.900000	
max	14.200000	1.100000	1.660000	65.800000	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
count	4898.000000	4898.000000	4898.000000	4898.000000	
mean	0.045772	35.308085	138.360657	0.994027	
std	0.021848	17.007137	42.498065	0.002991	
min	0.009000	2.000000	9.000000	0.987110	
25%	0.036000	23.000000	108.000000	0.991723	
50%	0.043000	34.000000	134.000000	0.993740	
75%	0.050000	46.000000	167.000000	0.996100	
max	0.346000	289.000000	440.000000	1.038980	

	pH	sulphates	alcohol	quality
count	4898.000000	4898.000000	4898.000000	4898.000000
mean	3.188267	0.489847	10.514267	5.877909
std	0.151001	0.114126	1.230621	0.885639
min	2.720000	0.220000	8.000000	3.000000
25%	3.090000	0.410000	9.500000	5.000000
50%	3.180000	0.470000	10.400000	6.000000
75%	3.280000	0.550000	11.400000	6.000000
max	3.820000	1.080000	14.200000	9.000000

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

```
# Langkah 2: Analisis Data Eksplorasi
# Memeriksa nilai yang hilang
print("\nNilai yang Hilang:")
print(data.isnull().sum())

# Penjelasan:
# - Langkah ini dilakukan untuk memastikan bahwa tidak ada data yang hilang, karena data yang hilang dapat memengaruhi model.
```



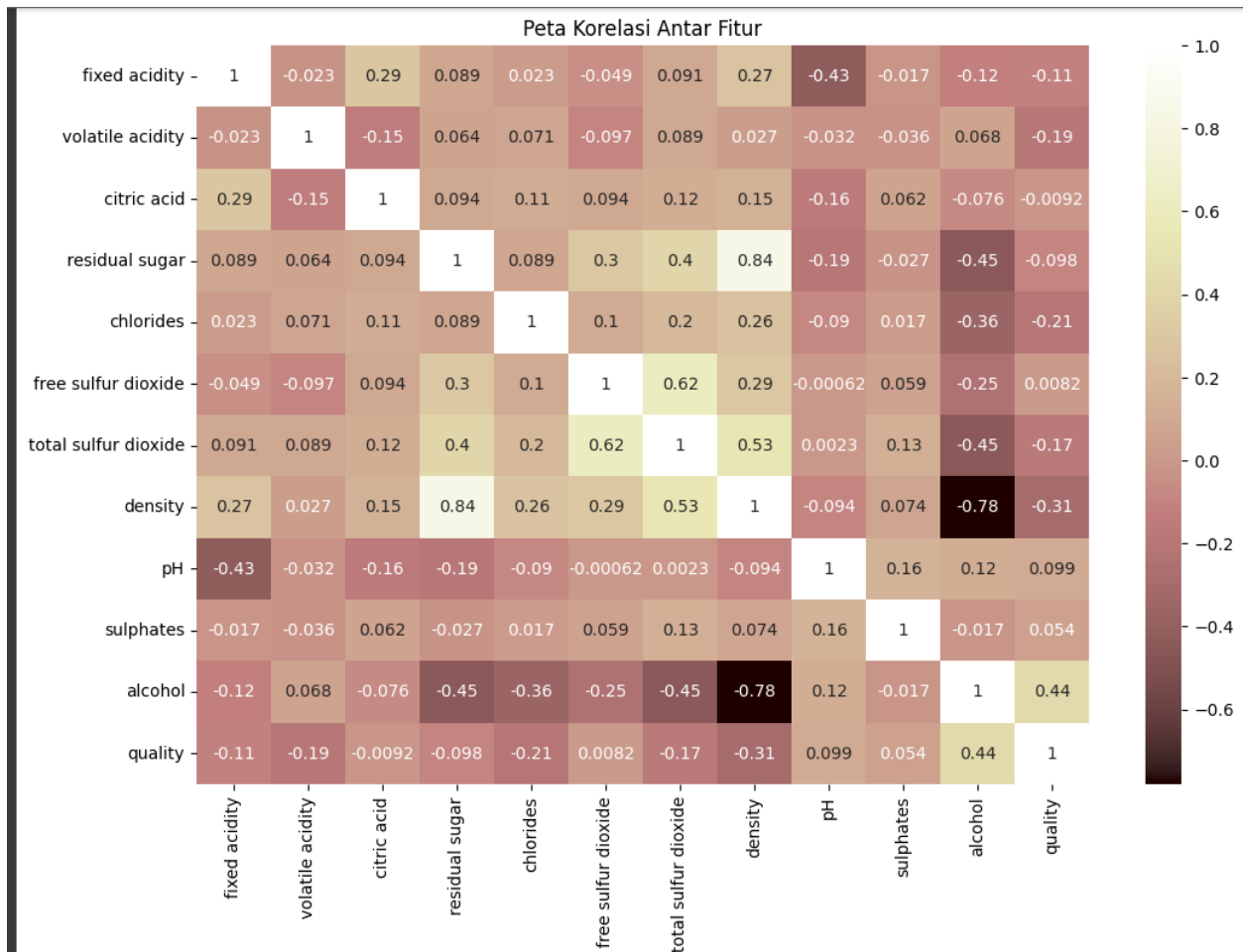
Nilai yang Hilang:

fixed acidity	0
volatile acidity	0
citric acid	0
residual sugar	0
chlorides	0
free sulfur dioxide	0
total sulfur dioxide	0
density	0
pH	0
sulphates	0
alcohol	0
quality	0
dtype:	int64

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

```
# Visualisasi korelasi antar fitur
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), annot=True, cmap='pink')
plt.title("Peta Korelasi Antar Fitur")
plt.show()
```

# Penjelasan:  
 # - Peta korelasi membantu kita memahami hubungan antar fitur.  
 # - Korelasi yang tinggi dapat mengindikasikan multikolinieritas, yang dapat memengaruhi model regresi.

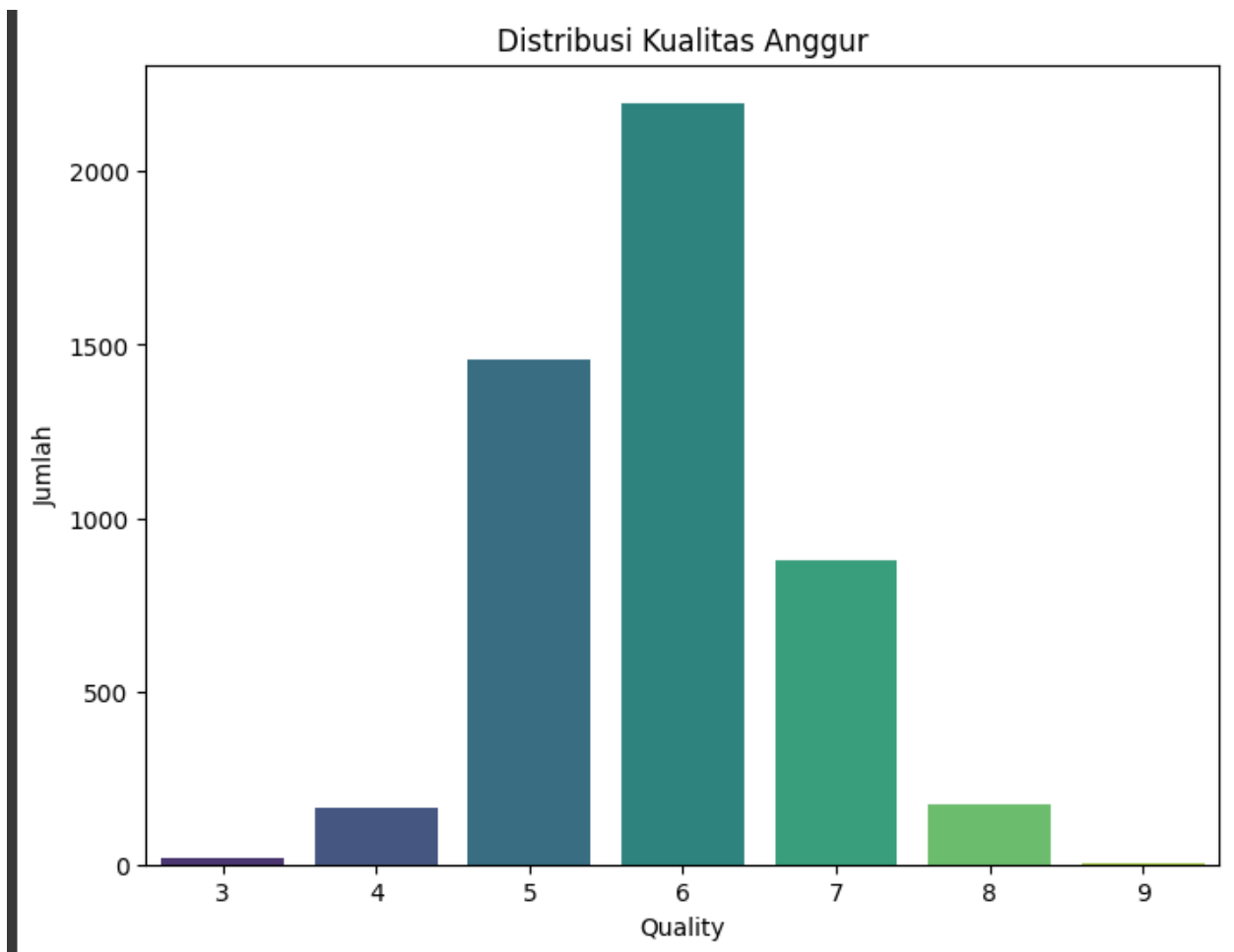


memvisualisasi korelasi antar fitur

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

```
# Distribusi variabel target (quality)
plt.figure(figsize=(8, 6))
sns.countplot(x='quality', data=data, palette='viridis')
plt.title("Distribusi Kualitas Anggur")
plt.xlabel("Quality")
plt.ylabel("Jumlah")
plt.show()

# Penjelasan:
# - Plot ini menunjukkan distribusi nilai kualitas dalam dataset.
# - Bias distribusi dapat memengaruhi pemilihan model, terutama jika data tidak seimbang.
```



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

```

# Langkah 3: Feature Engineering
# Mengubah variabel target menjadi biner (kualitas >= 6 dianggap "baik")
X = data.drop('quality', axis=1)
y = (data['quality'] >= 6).astype(int)

# Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Penjelasan:
# - Data dibagi menjadi data latih dan uji untuk evaluasi model yang akurat.
# - Stratifikasi memastikan distribusi variabel target tetap sama di kedua subset.

```

```

# Langkah 4: Definisi pipeline untuk setiap model
pipelines = {
    'LogisticRegression': Pipeline([
        ('scaler', StandardScaler()), # Scaling data
        ('model', LogisticRegression())
    ]),
    'DecisionTree': Pipeline([
        ('model', DecisionTreeClassifier(random_state=42))
    ]),
    'KNeighbors': Pipeline([
        ('scaler', StandardScaler()), # Scaling data
        ('model', KNeighborsClassifier())
    ]),
    'XGBoost': Pipeline([
        ('model', XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42))
    ])
}

# Penjelasan:
# - Pipeline mempermudah alur pemrosesan data dan pelatihan model.
# - Scaling diterapkan pada model yang sensitif terhadap skala (Logistic Regression, k-NN).

```

- 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).

```
[ ] # Langkah 5: Definisi parameter grid untuk hyperparameter tuning
param_grids = {
    'LogisticRegression': {
        'model__C': [0.1, 1, 10]
    },
    'DecisionTree': {
        'model__max_depth': [3, 5, 10],
        'model__min_samples_split': [2, 5, 10]
    },
    'KNeighbors': {
        'model__n_neighbors': [3, 5, 7],
        'model__weights': ['uniform', 'distance']
    },
    'XGBoost': {
        'model__n_estimators': [50, 100, 200],
        'model__learning_rate': [0.01, 0.1, 0.2],
        'model__max_depth': [3, 5, 7]
    }
}
```

```
[ ] # Langkah 6: Pelatihan model dan evaluasi
results = {}

for name, pipeline in pipelines.items():
    print(f"Melatih model: {name}")
    grid = GridSearchCV(pipeline, param_grids[name], cv=5, scoring='accuracy', n_jobs=-1)
    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)
    }

# Penjelasan:
# - GridSearchCV melakukan pencarian kombinasi terbaik dari hyperparameter untuk setiap model.
# - Skor akurasi digunakan sebagai metrik evaluasi.
```

- 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
/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,
Melatih model: XGBoost
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning: [16:58:06] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

warnings.warn(msg, UserWarning)

```



```

# Langkah 7: Menampilkan hasil
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)

# Penjelasan Output:
# - 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.

```

```

Model: LogisticRegression
Parameter Terbaik: {'model__C': 0.1}
Akurasi: 0.7438775510204082
Laporan Klasifikasi:

```

	precision	recall	f1-score	support
0	0.66	0.49	0.56	328
1	0.77	0.87	0.82	652
accuracy			0.74	980
macro avg	0.72	0.68	0.69	980
weighted avg	0.73	0.74	0.73	980

```

Confusion Matrix:
[[160 168]
 [ 83 569]]
-----
Model: DecisionTree
Parameter Terbaik: {'model__max_depth': 10, 'model__min_samples_split': 2}
Akurasi: 0.753061224489796
Laporan Klasifikasi:

```

	precision	recall	f1-score	support
0	0.64	0.61	0.62	328
1	0.81	0.83	0.82	652
accuracy			0.75	980
macro avg	0.72	0.72	0.72	980
weighted avg	0.75	0.75	0.75	980

```

Confusion Matrix:
[[199 129]
 [113 539]]
-----
Model: KNeighbors
Parameter Terbaik: {'model__n_neighbors': 7, 'model__weights': 'distance'}
Akurasi: 0.8214285714285714
Laporan Klasifikasi:

```

	precision	recall	f1-score	support
0	0.76	0.69	0.72	328
1	0.85	0.89	0.87	652

```

1      0.85      0.85      0.87      0.92
accuracy      0.82      980
macro avg      0.80      0.79      0.79      980
weighted avg      0.82      0.82      0.82      980

Confusion Matrix:
[[225 103]
 [ 72 580]]
-----
Model: XGBoost
Parameter Terbaik: {'model__learning_rate': 0.2, 'model__max_depth': 7, 'model__n_estimators': 200}
Akurasi: 0.8224489795918367
Laporan Klasifikasi:
      precision    recall  f1-score   support

     0       0.75       0.71       0.73       328
     1       0.86       0.88       0.87       652

accuracy      0.82      980
macro avg      0.80      0.79      0.80      980
weighted avg      0.82      0.82      0.82      980

Confusion Matrix:
[[233  95]
 [ 79 573]]
-----

```

- 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.