ARTIFICIAL INTELLIGENCE
KLASIFIKASI DAN REGRESI

Mata Kuliah	: Artificial intelligence
Dosen	: Dr. Kurnianingsih, S.T., M.T

Disusun Oleh:

Nama	: Danu Alamsyah Putra
Nim	: 4.33.23.1.06
Kelas	: TI-2B

JURUSAN TEKNIK ELEKTRO
PRODI TEKNOLOGI REKAYASA KOMPUTER
POLITEKNIK NEGERI SEMARANG
2024

1. Klasifikasi

Import Library

In [48]:
```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score, KFold, StratifiedKFold
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import  VotingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.pipeline import make_pipeline
import seaborn as sns
import matplotlib.pyplot as plt
```

Mengimpor library yang diperlukan untuk manipulasi data, preprocessing, pelatihan model, evaluasi, dan visualisasi.

EDA

In [49]:
```
#Exploratory Data Analysis
data = pd.read_csv('updated_pollution_dataset.csv')
data.head()
```

Out[49]:

	Temperature	Humidity	PM2.5	PM10	NO2	SO2	CO	Proximity_to_Industrial_Areas	Population_Density	Air Quality
0	29.8	59.1	5.2	17.9	18.9	9.2	1.72	6.3	319	Moderate
1	28.3	75.6	2.3	12.2	30.8	9.7	1.64	6.0	611	Moderate
2	23.1	74.7	26.7	33.8	24.4	12.6	1.63	5.2	619	Moderate
3	27.1	39.1	6.1	6.3	13.5	5.3	1.15	11.1	551	Good
4	26.5	70.7	6.9	16.0	21.9	5.6	1.01	12.7	303	Good

In [51]:
```
# Mengisi missing values dengan modus
for col in data.select_dtypes(include="object"):
    data[col].fillna(data[col].mode()[0], inplace=True)

# Define the mapping for air quality categories to numeric values
air_quality_mapping = {
    "Good": 3,
    "Moderate": 2,
    "Hazardous": 1,
    "Poor": 0
}

# Apply the mapping to the Air Quality column
data['Air Quality'] = data['Air Quality'].map(air_quality_mapping)

# Display the first few rows to verify the changes
data.head()
```

Out[51]:

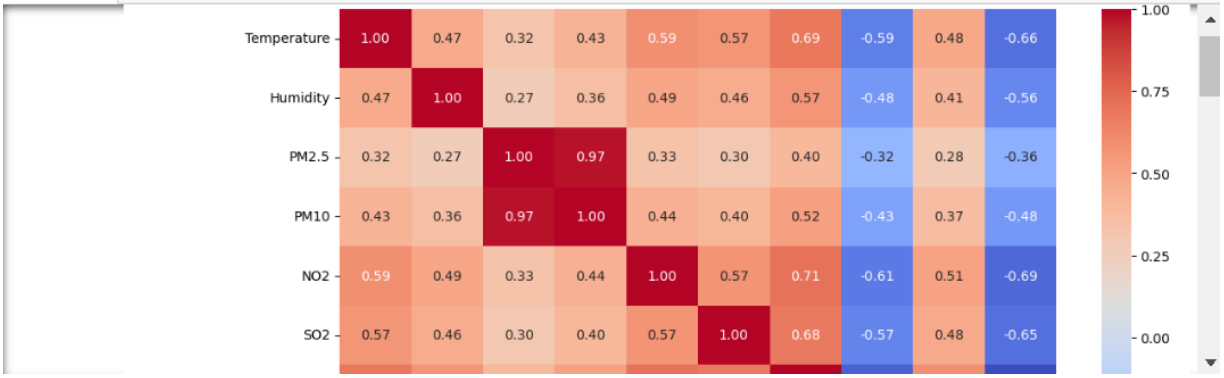
	Temperature	Humidity	PM2.5	PM10	NO2	SO2	CO	Proximity_to_Industrial_Areas	Population_Density	Air Quality
0	29.8	59.1	5.2	17.9	18.9	9.2	1.72	6.3	319	2
1	28.3	75.6	2.3	12.2	30.8	9.7	1.64	6.0	611	2
2	23.1	74.7	26.7	33.8	24.4	12.6	1.63	5.2	619	2
3	27.1	39.1	6.1	6.3	13.5	5.3	1.15	11.1	551	3
4	26.5	70.7	6.9	16.0	21.9	5.6	1.01	12.7	303	3

menampilkan beberapa baris pertama dari sebuah DataFrame, pada Column “Air Quality” yang sebelumnya non-numerik diganti angka numerik.

In [50]:
```
print("Data Preview:")
print(data.head())
print("----------------------------------------")
print("\nDataset Info:")
print(data.info())
print("----------------------------------------")
print("\nMissing Values:")
print(data.isnull().sum())
print("----------------------------------------")
print("\nTABEL ROW, KOLOM:")
print(data.shape)
```

```
Data Preview:
   Temperature  Humidity  PM2.5  PM10   NO2   SO2    CO  \
0         29.8      59.1    5.2  17.9  18.9   9.2  1.72
1         28.3      75.6    2.3  12.2  30.8   9.7  1.64
2         23.1      74.7   26.7  33.8  24.4  12.6  1.63
3         27.1      39.1    6.1   6.3  13.5   5.3  1.15
4         26.5      70.7    6.9  16.0  21.9   5.6  1.01

   Proximity_to_Industrial_Areas  Population_Density Air Quality
0                            6.3                 319    Moderate
1                            6.0                 611    Moderate
2                            5.2                 619    Moderate
3                           11.1                 551        Good
4                           12.7                 303        Good
----------------------------------------

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 10 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Temperature                    5000 non-null   float64
 1   Humidity                       5000 non-null   float64
 2   PM2.5                          5000 non-null   float64
 3   PM10                           5000 non-null   float64
 4   NO2                            5000 non-null   float64
 5   SO2                            5000 non-null   float64
 6   CO                             5000 non-null   float64
 7   Proximity_to_Industrial_Areas  5000 non-null   float64
 8   Population_Density             5000 non-null   int64
 9   Air Quality                    5000 non-null   object
dtypes: float64(8), int64(1), object(1)
memory usage: 390.8+ KB
None
----------------------------------------
Missing Values:
Temperature                    0
Humidity                       0
PM2.5                          0
PM10                           0
NO2                            0
SO2                            0
CO                             0
Proximity_to_Industrial_Areas  0
Population_Density             0
Air Quality                    0
dtype: int64
----------------------------------------

TABEL ROW, KOLOM:
(5000, 10)
```

In [52]:
```
# Correlation analysis (Pearson for linear relationships)
correlation_matrix = data.corr(method='pearson')
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()

# Pairplot for numerical features
sns.pairplot(data, diag_kind="kde")
plt.suptitle("Pairplot of Features", y=1.02)
plt.show()

# Encoding kategorik (jika ada)
categorical_columns = data.select_dtypes(include=['object']).columns
data = pd.get_dummies(data, columns=categorical_columns, drop_first=True)
```



Tujuan:

- Memuat dataset dan melakukan eksplorasi awal untuk memahami struktur data, melihat pratinjau data, memeriksa nilai yang hilang, dan mendapatkan informasi dasar tentang dataset.
- Mengonversi variabel kategorikal menjadi format numerik menggunakan one-hot encoding.
- Menghitung dan memvisualisasikan matriks korelasi untuk memahami hubungan antara fitur-fitur.
- Memvisualisasikan hubungan pasangan dalam dataset, diwarnai berdasarkan variabel target Air Quality.

Seleksi Fitur

In [56]:
```
# Seleksi fitur
correlation_threshold = 0.5
# ganti 0,5

correlations = correlation_matrix['Air Quality']
selected_features = correlations[abs(correlations) > correlation_threshold].index.tolist()

# Pastikan target tidak termasuk dalam fitur
selected_features.remove('Air Quality')

# Fitur yang didrop
dropped_features = correlations[abs(correlations) <= correlation_threshold].index.tolist()

print("\n--- Fitur yang Dipilih Berdasarkan Korelasi ---")
print(selected_features)

print("\n--- Fitur yang Didrop ---")
print(dropped_features)

# Pisahkan fitur dan target
X = data[selected_features]
y = data['Air Quality']

# Standarisasi fitur numerik
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Membagi data menjadi training dan testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state=42, stratify=y)
```

```
--- Fitur yang Dipilih Berdasarkan Korelasi ---
['Temperature', 'Humidity', 'NO2', 'SO2', 'CO', 'Proximity_to_Industrial_Areas', 'Population_Density']

--- Fitur yang Didrop ---
['PM2.5', 'PM10']
```

Memilih fitur-fitur yang memiliki korelasi tinggi dengan variabel target dan memisahkan fitur dan target untuk pelatihan model.

Model

In [57]:
```
# Model evaluation
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define models
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000, solver='liblinear', C=0.1, class_weight='balanced'),
    "SVM": SVC(),
    "Decision Tree": DecisionTreeClassifier(),
    "Naive Bayes": GaussianNB(),
    "knn": KNeighborsClassifier()
}

# Train and evaluate each model with cross-validation
results = {}
kf = KFold(n_splits=10, shuffle=True, random_state=42)
for name, model in models.items():
    pipeline = make_pipeline(scaler, model)
    cv_scores = cross_val_score(pipeline, X, y, cv=kf, scoring='accuracy')
    pipeline.fit(X_train, y_train)
    y_pred = pipeline.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    results[name] = {
        'accuracy': accuracy,
        'cv_mean': cv_scores.mean(),
        'cv_std': cv_scores.std()
    }
    print(f"\n{name} Accuracy: {accuracy:.2f}")
    print(classification_report(y_test, y_pred))
    print(f"{name} Cross-Validation Accuracy (Mean): {cv_scores.mean():.2f}")
    print(f"{name} Cross-Validation Accuracy (Standard Deviation): {cv_scores.std():.2f}")

# Display average cross-validation accuracy and standard deviation for all models
cv_means = [result['cv_mean'] for result in results.values()]
cv_stds = [result['cv_std'] for result in results.values()]
average_cv_mean = np.mean(cv_means)
average_cv_std = np.mean(cv_stds)

print(f"Average Cross-Validation Accuracy (Mean) for all models: {average_cv_mean:.2f}")
print(f"Average Cross-Validation Accuracy (Standard Deviation) for all models: {average_cv_std:.2f}")
```

```
Logistic Regression Accuracy: 0.86
              precision    recall  f1-score   support

           0       0.73      0.63      0.68       186
           1       0.79      0.83      0.81       111
           2       0.84      0.82      0.83       294
           3       0.94      1.00      0.97       409

    accuracy                           0.86      1000
   macro avg       0.82      0.82      0.82      1000
weighted avg       0.85      0.86      0.86      1000

Logistic Regression Cross-Validation Accuracy (Mean): 0.86
Logistic Regression Cross-Validation Accuracy (Standard Deviation): 0.01

SVM Accuracy: 0.94
              precision    recall  f1-score   support

           0       0.83      0.88      0.85       186
           1       0.87      0.79      0.83       111
           2       0.97      0.96      0.96       294
           3       1.00      1.00      1.00       409

    accuracy                           0.94      1000
   macro avg       0.92      0.91      0.91      1000
weighted avg       0.94      0.94      0.94      1000

SVM Cross-Validation Accuracy (Mean): 0.94
SVM Cross-Validation Accuracy (Standard Deviation): 0.01

Decision Tree Accuracy: 0.92
              precision    recall  f1-score   support

           0       0.81      0.79      0.80       186
           1       0.77      0.89      0.83       111
           2       0.96      0.91      0.94       294
           3       1.00      1.00      1.00       409

    accuracy                           0.92      1000
   macro avg       0.88      0.90      0.89      1000
weighted avg       0.93      0.92      0.92      1000

Decision Tree Cross-Validation Accuracy (Mean): 0.92
Decision Tree Cross-Validation Accuracy (Standard Deviation): 0.01

Naive Bayes Accuracy: 0.94
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       186
           1       0.88      0.80      0.84       111
           2       0.97      0.96      0.96       294
           3       1.00      1.00      1.00       409

    accuracy                           0.94      1000
   macro avg       0.92      0.91      0.92      1000
weighted avg       0.95      0.94      0.94      1000

Naive Bayes Cross-Validation Accuracy (Mean): 0.94
Naive Bayes Cross-Validation Accuracy (Standard Deviation): 0.01
```

```
knn Accuracy: 0.94
              precision    recall  f1-score   support

           0       0.82      0.87      0.84       186
           1       0.90      0.75      0.82       111
           2       0.95      0.96      0.96       294
           3       0.99      1.00      1.00       409

    accuracy                           0.94      1000
   macro avg       0.92      0.90      0.90      1000
weighted avg       0.94      0.94      0.94      1000

knn Cross-Validation Accuracy (Mean): 0.93
knn Cross-Validation Accuracy (Standard Deviation): 0.01
Average Cross-Validation Accuracy (Mean) for all models: 0.92
Average Cross-Validation Accuracy (Standard Deviation) for all models: 0.01
```

In [58]:
```python
# Inisialisasi model
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Support Vector Machine': SVC(probability=True),
    'Decision Tree': DecisionTreeClassifier(),
    'Naive Bayes': GaussianNB()
}

# Evaluasi model dengan cross-validation
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
all_results = pd.DataFrame(columns=['Model', 'Accuracy'])

for name, model in models.items():
    scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
    print(f'{name} Hasil Cross-Validation (10 Folds): {scores.mean():.2f} ± {scores.std():.2f}')
    all_results = pd.concat([all_results, pd.DataFrame({'Model': [name], 'Accuracy': [scores.mean()]})], ignore_index=True)
```

```
Logistic Regression Hasil Cross-Validation (10 Folds): 0.93 ± 0.01
K-Nearest Neighbors Hasil Cross-Validation (10 Folds): 0.93 ± 0.01
Support Vector Machine Hasil Cross-Validation (10 Folds): 0.94 ± 0.01
Decision Tree Hasil Cross-Validation (10 Folds): 0.91 ± 0.01
Naive Bayes Hasil Cross-Validation (10 Folds): 0.94 ± 0.01
```

Membagi data menjadi set pelatihan dan pengujian, mendefinisikan berbagai model klasifikasi, mengevaluasi setiap model menggunakan cross-validation, dan mencetak hasilnya.

Ensemble Learning

In [59]:
```python
# Membuat ensemble model dengan Soft Voting
soft_voting_model = VotingClassifier(
    estimators=[
        ('logreg', LogisticRegression(max_iter=1000, solver='liblinear', C=0.1, class_weight='balanced')),
        ('svc', SVC(probability=True)),
        ('dt', DecisionTreeClassifier())
    ],
    voting='soft'  # Menggunakan Soft Voting
)

# Cross-validation untuk Soft Voting Ensemble
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
cv_scores_soft_voting = cross_val_score(soft_voting_model, X_train, y_train, cv=cv, scoring='accuracy')

# Melatih ensemble model
soft_voting_model.fit(X_train, y_train)
y_pred_soft_voting = soft_voting_model.predict(X_test)

# Evaluasi performa ensemble model
print("\nEvaluasi Soft Voting Ensemble:")
print(classification_report(y_test, y_pred_soft_voting))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_soft_voting))
soft_voting_accuracy = accuracy_score(y_test, y_pred_soft_voting)
print(f'Soft Voting Ensemble Accuracy on Test Set: {soft_voting_accuracy:.2f}')

# Menambahkan hasil Cross-Validation ke output
print(f"\nEnsemble Model Cross-Validation Accuracy (Mean): {cv_scores_soft_voting.mean():.2f}")
print(f"Ensemble Model Cross-Validation Accuracy (Standard Deviation): {cv_scores_soft_voting.std():.2f}")

# Menampilkan hasil evaluasi semua model
print("\nHasil Evaluasi Model:")
print(all_results)
```

```
Evaluasi Soft Voting Ensemble:
              precision    recall  f1-score   support

           0       0.82      0.83      0.83       186
           1       0.81      0.86      0.83       111
           2       0.96      0.94      0.95       294
           3       1.00      1.00      1.00       409

    accuracy                           0.93      1000
   macro avg       0.90      0.90      0.90      1000
weighted avg       0.93      0.93      0.93      1000

Confusion Matrix:
 [[154  22  10   0]
 [ 16  95   0   0]
 [ 17   0 275   2]
 [  0   0   0 409]]
Soft Voting Ensemble Accuracy on Test Set: 0.93

Ensemble Model Cross-Validation Accuracy (Mean): 0.93
Ensemble Model Cross-Validation Accuracy (Standard Deviation): 0.01

Hasil Evaluasi Model:
                    Model  Accuracy
0     Logistic Regression   0.93225
1     K-Nearest Neighbors   0.92825
2  Support Vector Machine   0.94400
3           Decision Tree   0.91200
4             Naive Bayes   0.94175
```

2. Regresi

Import Library dan ambil dataset

In [10]:
```
# Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.feature_selection import mutual_info_regression
from sklearn.ensemble import GradientBoostingRegressor, VotingRegressor, StackingRegressor, BaggingRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import RidgeCV
import seaborn as sns
import matplotlib.pyplot as plt
```

In [11]:
```
# 1. Load Data
data = pd.read_csv('Water Quality Testing.csv')
```

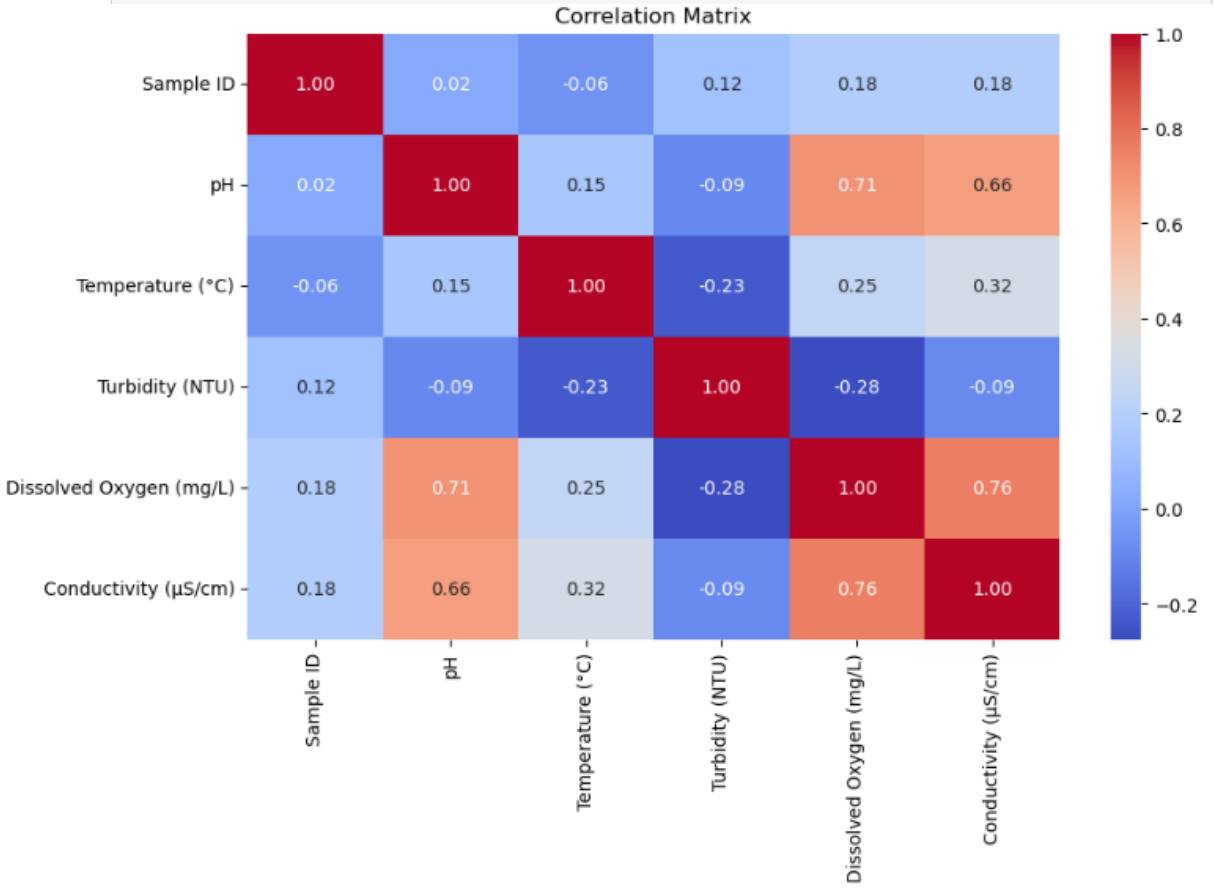
Mengimpor pustaka yang diperlukan untuk manipulasi data, prapemrosesan, pelatihan model, dan evaluasi. Memuat dataset Water Quality Tetsting.csv ke dalam DataFrame df.

EDA

In [12]:
```
# 2. EDA
print("Informasi Dataset:")
print(data.info())  # Informasi dataset
print("\nSampel Data:")
print(data.head())  # Menampilkan data sampel
print("\nJumlah Nilai Kosong:")
print(data.isnull().sum())  # Jumlah missing values
print("\nTABEL ROW, KOLOM:")
print(data.shape)
```

```
Informasi Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 6 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Sample ID               500 non-null    int64
 1   pH                      500 non-null    float64
 2   Temperature (°C)        500 non-null    float64
 3   Turbidity (NTU)         500 non-null    float64
 4   Dissolved Oxygen (mg/L) 500 non-null    float64
 5   Conductivity (µS/cm)    500 non-null    int64
dtypes: float64(4), int64(2)
memory usage: 23.6 KB
None

Sampel Data:
   Sample ID    pH  Temperature (°C)  Turbidity (NTU)  \
0          1  7.25              23.1              4.5
1          2  7.11              22.3              5.1
2          3  7.03              21.5              3.9
3          4  7.38              22.9              3.2
4          5  7.45              20.7              3.8

   Dissolved Oxygen (mg/L)  Conductivity (µS/cm)
0                      7.8                   342
1                      6.2                   335
2                      8.3                   356
3                      9.5                   327
4                      8.1                   352

Jumlah Nilai Kosong:
Sample ID                 0
pH                        0
Temperature (°C)          0
Turbidity (NTU)           0
Dissolved Oxygen (mg/L)   0
Conductivity (µS/cm)      0
dtype: int64

TABEL ROW, KOLOM:
(500, 6)
```

In [13]:
```
# Menghapus nilai kosong
data_cleaned = data[numerical_cols + [target_col]].dropna()

# Pairplot for numerical features
sns.pairplot(data, diag_kind="kde")
plt.suptitle("Pairplot of Features", y=1.02)
plt.show()

# Correlation Matrix
correlation_matrix = data.corr()
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```



Melakukan eksplorasi awal data untuk memahami struktur data, melihat pratinjau data, memeriksa nilai yang hilang, dan mendapatkan informasi dasar tentang dataset. Mengonversi variabel kategorikal menjadi format numerik menggunakan Label Encoding. Memvisualisasikan matriks korelasi dan hubungan pasangan antar fitur.

Seleksi Fitur

In [15]:
```
# Seleksi Fitur Berdasarkan Korelasi Pearson
# Menghitung korelasi Pearson dengan kolom 'Conductivity (µS/cm)' sebagai target
pearson_corr = data_cleaned.drop(columns=['Conductivity (µS/cm)']).corrwith(data_cleaned['Conductivity (µS/cm)'], method='pearson

# Menampilkan hasil korelasi
print("Korelasi Pearson:\n", pearson_corr)

# Menentukan ambang batas (threshold)
threshold = 0.3  # Korelasi minimum untuk memilih fitur

# Memfilter fitur berdasarkan ambang batas korelasi Pearson
pearson_filtered = pearson_corr[pearson_corr >= threshold]

# Membuat DataFrame untuk korelasi Pearson
pearson_df = pd.DataFrame({'Fitur': pearson_filtered.index, 'Pearson': pearson_filtered.values})

# Menampilkan fitur yang dipilih
print(f"\nFitur yang dipilih berdasarkan ambang batas korelasi Pearson ({threshold}):")
print(pearson_df)

# Mengidentifikasi fitur yang di-drop
dropped_features = pearson_corr[pearson_corr < threshold].index.tolist()
print(f"\nFitur yang di-drop berdasarkan ambang batas korelasi Pearson ({threshold}):")
print(dropped_features)

# Memilih fitur yang relevan dari data_cleaned berdasarkan korelasi Pearson
X_selected = data_cleaned[pearson_df['Fitur']]
```

```
Korelasi Pearson:
 Dissolved Oxygen (mg/L)    0.760700
pH                          0.664232
Temperature (°C)            0.317573
Sample ID                   0.184919
Turbidity (NTU)            -0.087372
dtype: float64

Fitur yang dipilih berdasarkan ambang batas korelasi Pearson (0.3):
                      Fitur   Pearson
0  Dissolved Oxygen (mg/L)  0.760700
1                       pH  0.664232
2         Temperature (°C)  0.317573

Fitur yang di-drop berdasarkan ambang batas korelasi Pearson (0.3):
['Sample ID', 'Turbidity (NTU)']
```

Menghitung matriks korelasi Pearson antara fitur dan target. Memilih fitur yang memiliki korelasi di atas ambang batas tertentu (0.3). Memastikan variabel target tidak termasuk dalam fitur yang dipilih. Membuat DataFrame baru dengan fitur yang dipilih.

Evaluasi Model Individu & Model Ensemble

In [16]:
```
# 4. Data Preparation
# Split data
X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, random_state=42)

# Define models
models = {
    "Linear Regression": LinearRegression(),
    "Polynomial Regression": make_pipeline(PolynomialFeatures(degree=2), LinearRegression()),
    "Lasso Regression": Lasso(alpha=0.01),
    "Ridge Regression": Ridge(alpha=1),
    "Decision Tree Regression": DecisionTreeRegressor(random_state=42)
}

# K-Fold cross-validation with 10 folds
kf = KFold(n_splits=10, shuffle=True, random_state=42)

# Train and evaluate each model
for name, model in models.items():
    # R-squared
    model.fit(X_train, y_train)
    r_squared = model.score(X_test, y_test)
    print(f"{name} R-squared: {r_squared:.4f}")

    # Cross-validated RMSE
    ensemble_scores = cross_val_score(model, X_train, y_train, cv=kf, scoring='neg_mean_squared_error')
    ensemble_rmse = np.sqrt(-ensemble_scores.mean())
    print(f"{name} Cross-validated RMSE: {ensemble_rmse:.4f}")
    print('------')

# Ensemble Learning: Soft Voting with 5 models
ensemble = VotingRegressor(estimators=[
    ('LR', LinearRegression()),
    ('PR', make_pipeline(PolynomialFeatures(degree=2), LinearRegression())),
    ('Lasso', Lasso(alpha=0.01)),
    ('Ridge', Ridge(alpha=1)),
    ('DT', DecisionTreeRegressor(random_state=42))
])

# Train and evaluate ensemble model
ensemble.fit(X_train, y_train)
ensemble_r_squared = ensemble.score(X_test, y_test)
print(f"Ensemble R-squared: {ensemble_r_squared:.4f}")

# Cross-validated RMSE for ensemble model
ensemble_scores = cross_val_score(ensemble, X_train, y_train, cv=kf, scoring='neg_mean_squared_error')
ensemble_rmse = np.sqrt(-ensemble_scores.mean())
print(f"Ensemble Cross-validated RMSE: {ensemble_rmse:.4f}")
```

```
Linear Regression R-squared: 0.5475
Linear Regression Cross-validated RMSE: 7.7252
------
Polynomial Regression R-squared: 0.6101
Polynomial Regression Cross-validated RMSE: 7.2904
------
Lasso Regression R-squared: 0.5490
Lasso Regression Cross-validated RMSE: 7.7252
------
Ridge Regression R-squared: 0.5539
Ridge Regression Cross-validated RMSE: 7.7472
------
Decision Tree Regression R-squared: 0.5472
Decision Tree Regression Cross-validated RMSE: 8.4131
------
Ensemble R-squared: 0.6184
Ensemble Cross-validated RMSE: 7.1314
```

Membagi data menjadi set pelatihan dan pengujian. Mendefinisikan berbagai model regresi. Melakukan evaluasi setiap model menggunakan cross-validation dengan 10 lipatan (folds). Melatih dan mengevaluasi model ensemble menggunakan Voting Regressor dengan 5 model regresi. Mencetak hasil evaluasi model individu dan model ensemble.