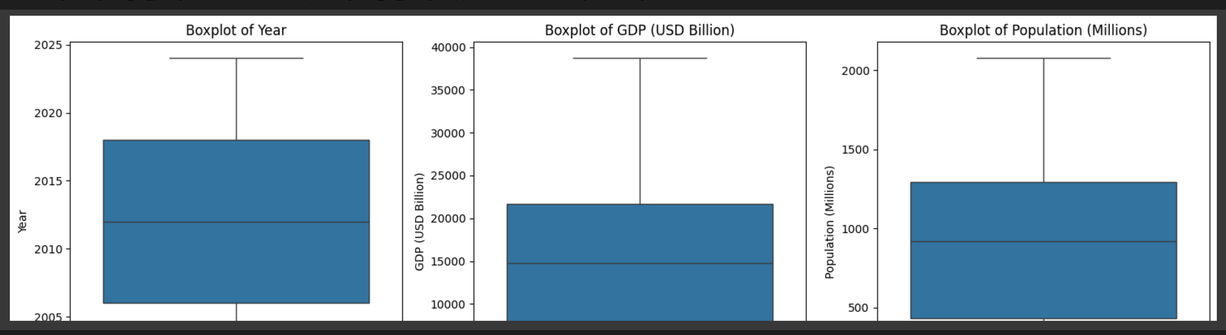
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.preprocessing import StandardScaler
from matplotlib import gridspec
```

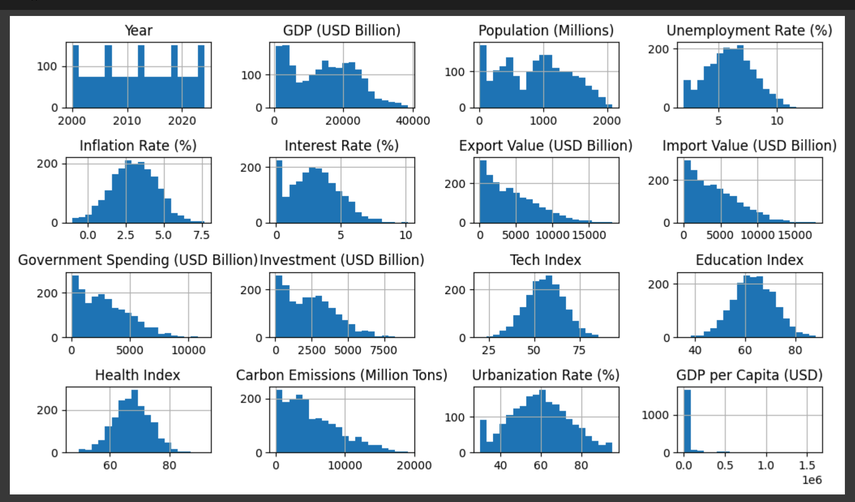
```
df = pd.read_csv('global_gdp_dataset.csv')
print(" Null values before handling:\n", df.isnull().sum())
df_before = df.copy()
for column in df.columns:
    if df[column].isnull().sum() > 0:
        if df[column].dtype == 'float64' or df[column].dtype == 'int64':
            df[column].fillna(df[column].mean(), inplace=True)
        else:
            df[column].fillna(df[column].mode()[0], inplace=True)
```

```
 Null values before handling:
 Year                              0
Country                           0
GDP (USD Billion)                 0
Population (Millions)             0
Unemployment Rate (%)             0
Inflation Rate (%)                0
Interest Rate (%)                 0
Export Value (USD Billion)        0
Import Value (USD Billion)        0
Government Spending (USD Billion) 0
Investment (USD Billion)          0
Tech Index                        0
Education Index                   0
Health Index                      0
Carbon Emissions (Million Tons)   0
Urbanization Rate (%)             0
GDP per Capita (USD)              0
dtype: int64
```

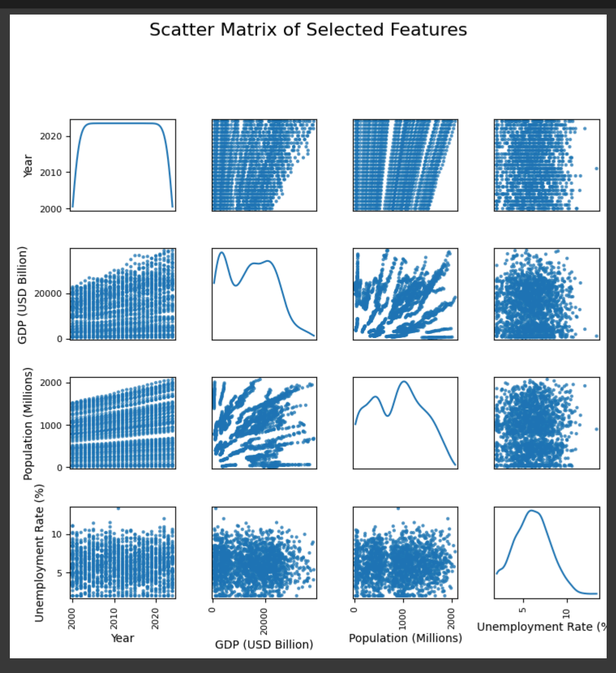
```
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns.tolist()
cols_to_plot = numeric_cols[:3]
plt.figure(figsize=(15, 5))
for i, col in enumerate(cols_to_plot):
    plt.subplot(1, 3, i + 1)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col}')
plt.tight_layout()
plt.show()
Q1 = df[cols_to_plot].quantile(0.25)
Q3 = df[cols_to_plot].quantile(0.75)
IQR = Q3 - Q1
df_clean = df[~((df[cols_to_plot] < (Q1 - 1.5 * IQR)) | (df[cols_to_plot] > (Q3 + 1.5 * IQR))).any(axis=1)]
```



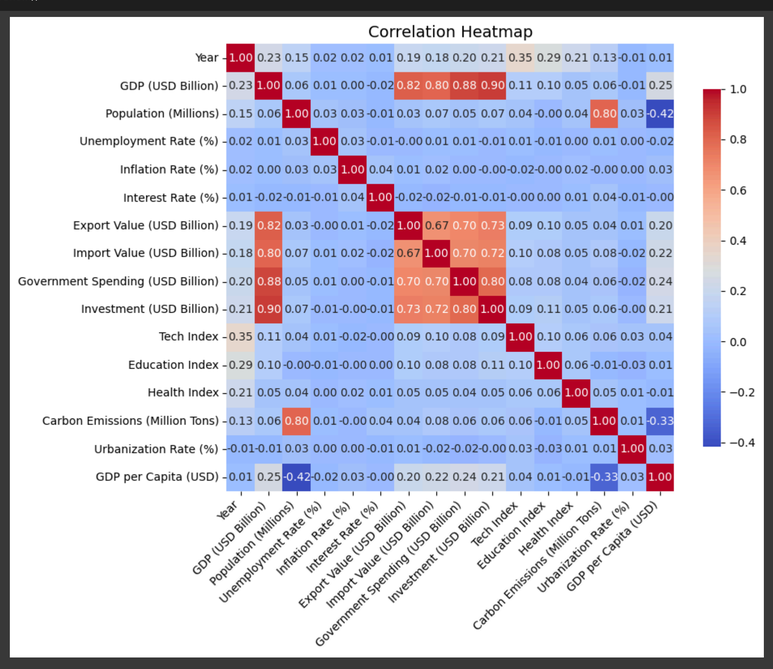

```
df[numeric_cols].hist(figsize=(10, 6), bins=20)
plt.tight_layout()
plt.show()
```




```
num_cols = df.select_dtypes(include=['int64', 'float64']).columns[:4]
scatter_matrix(df[num_cols], figsize=(8, 8), alpha=0.8, diagonal='kde')
plt.suptitle("Scatter Matrix of Selected Features", fontsize=16, y=1.02)
plt.tight_layout(pad=3.0)
plt.show()
```




```
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
plt.figure(figsize=(10, 8))
sns.heatmap(df[numeric_cols].corr(),
            annot=True,
            fmt=".2f",
            cmap='coolwarm',
            annot_kws={"size": 10},
            square=True,
            cbar_kws={"shrink": .8})
plt.title("Correlation Heatmap", fontsize=14)
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```




```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
training_df = pd.concat([X_train, y_train], axis=1)
testing_df = pd.concat([X_test, y_test], axis=1)
training_df.to_csv('training.csv', index=False)
```

```
testing_df.to_csv('testing.csv', index=False)
print(" Saved 'training.csv' and 'testing.csv' with 70:30 split.")
```

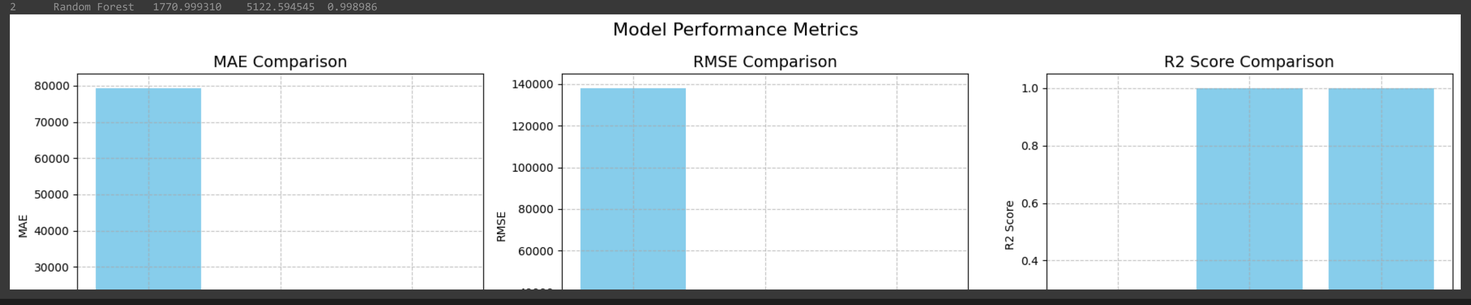
```
 Saved 'training.csv' and 'testing.csv' with 70:30 split.
```

```
train_df = pd.read_csv('training.csv')
test_df = pd.read_csv('testing.csv')
categorical_features = train_df.select_dtypes(include=['object']).columns
label_encoders = {}
for feature in categorical_features:
    label_encoders[feature] = LabelEncoder()
for feature in categorical_features:
    train_df[feature] = label_encoders[feature].fit_transform(train_df[feature])
for feature in categorical_features:
    test_df[feature] = label_encoders[feature].transform(test_df[feature])
X_train = train_df.iloc[:, :-1]
y_train = train_df.iloc[:, -1]
X_test = test_df.iloc[:, :-1]
y_test = test_df.iloc[:, -1]
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor()
}
print(" Model Performance Metrics:\n")
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)
    print(f"=== {name} ===")
    print(f"Accuracy (R²): {r2 * 100:.2f}%")
    print(f"MAE: {mae:.2f}")
    print(f"RMSE: {rmse:.2f}\n")
```

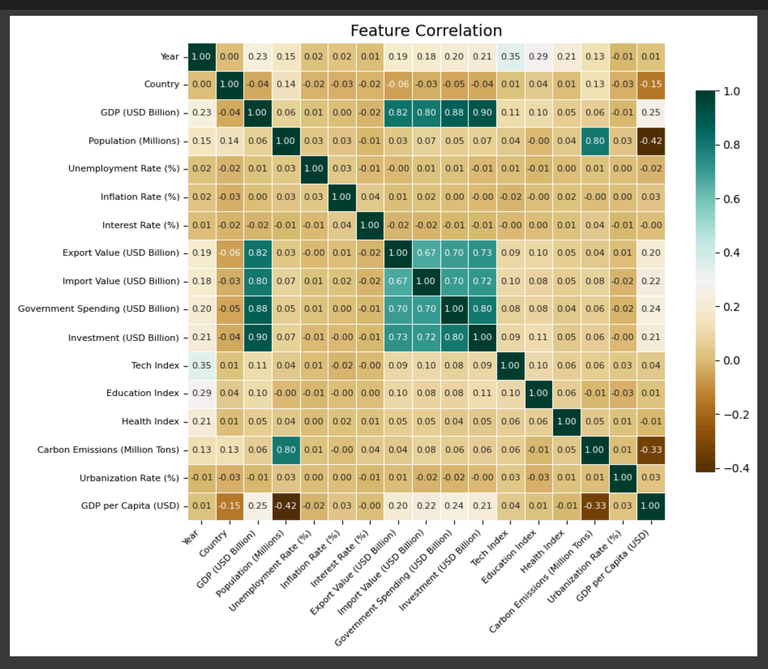
```
 Model Performance Metrics:

=== Linear Regression ===
Accuracy (R²): 26.41%
MAE: 79317.49
RMSE: 137982.37

=== Decision Tree ===
Accuracy (R²): 99.81%
MAE: 2164.77
RMSE: 6992.16

=== Random Forest ===
Accuracy (R²): 99.91%
MAE: 1769.28
RMSE: 4914.10
```

```
performance = []
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    r2 = r2_score(y_test, y_pred)
    performance.append({
        "Model": name,
        "MAE": mae,
        "RMSE": rmse,
        "R2 Score": r2
    })
performance_df = pd.DataFrame(performance)
print("Model Performance Summary:")
print(performance_df)
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
metrics = ["MAE", "RMSE", "R2 Score"]
x_ticks = range(len(performance_df["Model"]))
x_labels = performance_df["Model"]
for i, metric in enumerate(metrics):
    axes[i].bar(x_ticks, performance_df[metric], color='skyblue')
    axes[i].set_title(f'{metric} Comparison', fontsize=14)
    axes[i].set_xlabel("Model")
    axes[i].set_ylabel(metric)
    axes[i].grid(True, linestyle='--', alpha=0.7)
    axes[i].set_xticks(x_ticks)
    axes[i].set_xticklabels(x_labels, rotation=15)
plt.tight_layout()
plt.suptitle("Model Performance Metrics", fontsize=16, y=1.05)
plt.show()
```

```
Model Performance Summary:
               Model           MAE           RMSE  R2 Score
0  Linear Regression  79317.492942  137982.366611  0.264088
1      Decision Tree   2035.216199    7050.427688  0.998079
2      Random Forest   1770.999310    5122.594545  0.998986
```




```
df = pd.read_csv('global_gdp_dataset.csv')
categorical_cols = df.select_dtypes(include='object').columns
for col in categorical_cols:
    df[col] = LabelEncoder().fit_transform(df[col].astype(str))
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
plt.figure(figsize=(10, 8))
sns.heatmap(df[numeric_cols].corr(),
            annot=True,
            fmt=".2f",
            cmap='BrBG',
            square=True,
            annot_kws={"size": 8},
            cbar_kws={"shrink": .8},
            linewidths=0.5,
            linecolor='white')
plt.title("Feature Correlation", fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=8)
plt.yticks(fontsize=8)
plt.tight_layout()
plt.show()
```

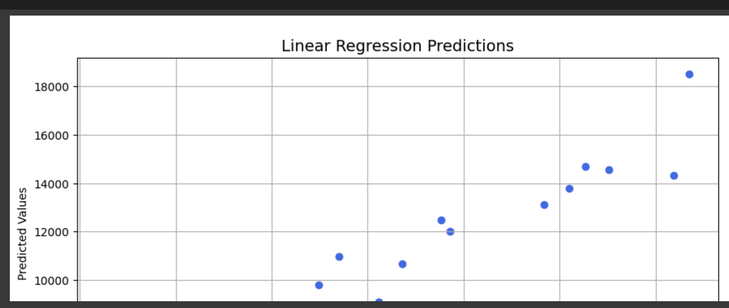



```
categorical_cols = df.select_dtypes(include='object').columns
for col in categorical_cols:
    df[col] = LabelEncoder().fit_transform(df[col].astype(str))
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
y_discretized = pd.qcut(y, q=3, labels=[0, 1, 2])
X_scaled = StandardScaler().fit_transform(X)
lda = LDA(n_components=2)
X_lda = lda.fit_transform(X_scaled, y_discretized)
plt.figure(figsize=(8, 6))
scatter = plt.scatter(X_lda[:, 0], X_lda[:, 1], c=y_discretized, cmap='viridis', alpha=0.7)
plt.xlabel('LD1')
plt.ylabel('LD2')
plt.title('LDA Projection of Dataset')
plt.colorbar(scatter, label='Target Classes')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
np.random.seed(42)
X = np.random.rand(100, 5) * 10000
y = X[:, 0] * 1.5 + X[:, 1] * 0.5 + np.random.normal(0, 1000, 100)
```

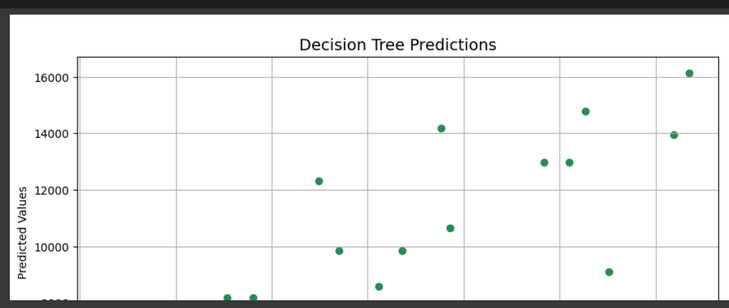
```
scaler = StandardScaler()
model = LinearRegression()
selected_features = [0, 1]
criteria = [
    ("All features, No scaling", X),
    ("All features, with scaling", scaler.fit_transform(X)),
    ("Selected Features, No scaling", X[:, selected_features]),
    ("Selected Features, with scaling", scaler.fit_transform(X[:, selected_features])),
]
results = []
fig = plt.figure(figsize=(18, 6))
gs = gridspec.GridSpec(1, 2, width_ratios=[2, 1.5])
ax0 = plt.subplot(gs[0])
ax1 = plt.subplot(gs[1])
for label, features in criteria:
    X_train, X_test, y_train, y_test = train_test_split(features, y, test_size=0.2, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results.append([
        label,
        round(mean_absolute_error(y_test, y_pred), 2),
        round(np.sqrt(mean_squared_error(y_test, y_pred)), 2),
        round(r2_score(y_test, y_pred), 2)
    ])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
ax0.scatter(y_test, y_pred, color="royalblue")
ax0.set_title("Linear Regression Predictions", fontsize=14)
ax0.set_xlabel("True Values")
ax0.set_ylabel("Predicted Values")
ax0.grid(True)
ax1.axis("off")
df_result = pd.DataFrame(results, columns=["Criteria", "MAE", "RMSE", "R2 Score"])
table = ax1.table(cellText=df_result.values, colLabels=df_result.columns, loc='center')
table.auto_set_font_size(False)
table.set_fontsize(9)
table.scale(1.4, 1.4)
ax1.set_title("Linear Regression Metrics", fontsize=14, fontweight="bold", pad=20)
plt.tight_layout(pad=3.0)
plt.show()
```




**Linear Regression Metrics**

| Criteria | MAE | RMSE | R2 Score |
|---|---|---|---|
| All features, No scaling | 776.47 | 987.9 | 0.93 |
| All features, with scaling | 776.47 | 987.9 | 0.93 |
| Selected Features, No scaling | 714.09 | 927.5 | 0.94 |
| Selected Features, with scaling | 714.09 | 927.5 | 0.94 |

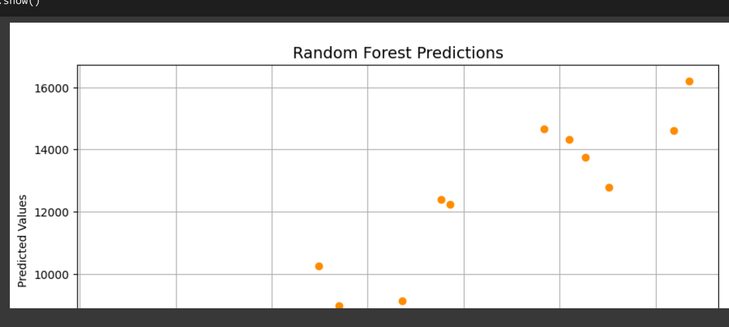
```
np.random.seed(42)
X = np.random.rand(100, 5) * 10000
y = X[:, 0] * 1.5 + X[:, 1] * 0.5 + np.random.normal(0, 1000, 100)
scaler = StandardScaler()
model = DecisionTreeRegressor()
selected_features = [0, 1]
criteria = [
    ("All features, No scaling", X),
    ("All features, with scaling", scaler.fit_transform(X)),
    ("Selected Features, No scaling", X[:, selected_features]),
    ("Selected Features, with scaling", scaler.fit_transform(X[:, selected_features])),
]
results = []
fig = plt.figure(figsize=(18, 6))
gs = gridspec.GridSpec(1, 2, width_ratios=[2, 1.5])
ax0 = plt.subplot(gs[0])
ax1 = plt.subplot(gs[1])
for label, features in criteria:
    X_train, X_test, y_train, y_test = train_test_split(features, y, test_size=0.2, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results.append([
        label,
        round(mean_absolute_error(y_test, y_pred), 2),
        round(np.sqrt(mean_squared_error(y_test, y_pred)), 2),
        round(r2_score(y_test, y_pred), 2)
    ])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
ax0.scatter(y_test, y_pred, color="seagreen")
ax0.set_title("Decision Tree Predictions", fontsize=14)
ax0.set_xlabel("True Values")
ax0.set_ylabel("Predicted Values")
ax0.grid(True)
ax1.axis("off")
df_result = pd.DataFrame(results, columns=["Criteria", "MAE", "RMSE", "R2 Score"])
table = ax1.table(cellText=df_result.values, colLabels=df_result.columns, loc='center')
table.auto_set_font_size(False)
table.set_fontsize(9)
table.scale(1.4, 1.4)
ax1.set_title("Decision Tree Metrics", fontsize=14, fontweight="bold", pad=20)
plt.tight_layout(pad=3.0)
plt.show()
```




**Decision Tree Metrics**

| Criteria | MAE | RMSE | R2 Score |
|---|---|---|---|
| All features, No scaling | 1712.4 | 2246.88 | 0.66 |
| All features, with scaling | 1691.17 | 2251.34 | 0.65 |
| Selected Features, No scaling | 1117.52 | 1541.72 | 0.84 |
| Selected Features, with scaling | 1012.94 | 1396.82 | 0.87 |

```
np.random.seed(42)
X = np.random.rand(100, 5) * 10000
y = X[:, 0] * 1.5 + X[:, 1] * 0.5 + np.random.normal(0, 1000, 100)
scaler = StandardScaler()
model = RandomForestRegressor(n_estimators=100, random_state=42)
selected_features = [0, 1]
criteria = [
    ("All features, No scaling", X),
    ("All features, with scaling", scaler.fit_transform(X)),
    ("Selected Features, No scaling", X[:, selected_features]),
    ("Selected Features, with scaling", scaler.fit_transform(X[:, selected_features])),
]
results = []
fig = plt.figure(figsize=(18, 6))
gs = gridspec.GridSpec(1, 2, width_ratios=[2, 1.5])
ax0 = plt.subplot(gs[0])
ax1 = plt.subplot(gs[1])
for label, features in criteria:
    X_train, X_test, y_train, y_test = train_test_split(features, y, test_size=0.2, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results.append([
        label,
        round(mean_absolute_error(y_test, y_pred), 2),
        round(np.sqrt(mean_squared_error(y_test, y_pred)), 2),
        round(r2_score(y_test, y_pred), 2)
    ])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
ax0.scatter(y_test, y_pred, color="darkorange")
ax0.set_title("Random Forest Predictions", fontsize=14)
ax0.set_xlabel("True Values")
ax0.set_ylabel("Predicted Values")
ax0.grid(True)
ax1.axis("off")
df_result = pd.DataFrame(results, columns=["Criteria", "MAE", "RMSE", "R2 Score"])
table = ax1.table(cellText=df_result.values, colLabels=df_result.columns, loc='center')
table.auto_set_font_size(False)
table.set_fontsize(9)
table.scale(1.4, 1.4)
ax1.set_title("Random Forest Metrics", fontsize=14, fontweight="bold", pad=20)
plt.tight_layout(pad=3.0)
plt.show()
```




**Random Forest Metrics**

| Criteria | MAE | RMSE | R2 Score |
|---|---|---|---|
| All features, No scaling | 967.17 | 1182.62 | 0.9 |
| All features, with scaling | 967.17 | 1182.62 | 0.9 |
| Selected Features, No scaling | 831.38 | 1107.12 | 0.92 |
| Selected Features, with scaling | 831.38 | 1107.12 | 0.92 |