```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
def mean_squared_error(y_true, y_pred, *, sample_weight=None, mul
```

Open in tab View source

Mean squared error regression loss.

Read more in the `User Guide <mean_squared_error>`.

Parameters

y_true : array-like of shape (n_samples,) or (n_samples, n_outputs)
    Ground truth (correct) target values.

y_pred : array-like of shape (n_samples,) or (n_samples, n_outputs)
    Estimated target values.

sample_weight : array-like of shape (n_samples,), default=None
    Sample weights

```
df = pd.read_csv("/content/Life Expectancy Data.csv")
```

```
print("Dataset Shape:", df.shape)
```

```
Dataset Shape: (2938, 22)
```

```
print(df.head())
```

```
       Country  Year      Status  Life expectancy   Adult Mortality  \
0  Afghanistan  2015  Developing             65.0            263.0
1  Afghanistan  2014  Developing             59.9            271.0
2  Afghanistan  2013  Developing             59.9            268.0
3  Afghanistan  2012  Developing             59.5            272.0
4  Afghanistan  2011  Developing             59.2            275.0

   infant deaths  Alcohol  percentage expenditure  Hepatitis B  Measles   ...  \
0             62     0.01               71.279624         65.0      1154  ...
1             64     0.01               73.523582         62.0       492  ...
2             66     0.01               73.219243         64.0       430  ...
3             69     0.01               78.184215         67.0      2787  ...
4             71     0.01                7.097109         68.0      3013  ...

   Polio  Total expenditure  Diphtheria    HIV/AIDS         GDP  Population  \
0    6.0               8.16         65.0        0.1  584.259210  33736494.0
1   58.0               8.18         62.0        0.1  612.696514    327582.0
2   62.0               8.13         64.0        0.1  631.744976  31731688.0
3   67.0               8.52         67.0        0.1  669.959000   3696958.0
4   68.0               7.87         68.0        0.1   63.537231   2978599.0

    thinness  1-19 years   thinness 5-9 years  \
0                   17.2                 17.3
1                   17.5                 17.5
2                   17.7                 17.7
3                   17.9                 18.0
4                   18.2                 18.2

   Income composition of resources  Schooling
0                            0.479       10.1
1                            0.476       10.0
2                            0.470        9.9
3                            0.463        9.8
4                            0.454        9.5

[5 rows x 22 columns]
```

```
df = df.dropna()
```

```
X = df.drop(["Life expectancy "], axis=1)
y = df["Life expectancy "]
```

```
X = pd.get_dummies(X, drop_first=True)
```

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

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
results = []
```

```
def evaluate_model(model, X_train, y_train, X_test, y_test, name="Model"):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print(f"\n{name} Performance:")
    print("R² Score:", round(r2, 4))
    print("MAE:", round(mae, 4))
    print("RMSE:", round(rmse, 4))

    results.append([name, r2, mae, rmse])
    return model


def linear_regression():
    return evaluate_model(LinearRegression(),X_train_scaled, y_train,X_test_scaled, y_test,"Linear Regression")


def polynomial_regression(degree=2):
    poly = PolynomialFeatures(degree=degree)
    X_train_poly = poly.fit_transform(X_train_scaled)
    X_test_poly = poly.transform(X_test_scaled)

    return evaluate_model(LinearRegression(),X_train_poly, y_train,X_test_poly, y_test,f"Polynomial Regression (deg={degree}


def decision_tree_regression():
    return evaluate_model(DecisionTreeRegressor(random_state=42),
                          X_train, y_train,
                          X_test, y_test,
                          "Decision Tree Regression")


def random_forest_regression(n_estimators=100):
    return evaluate_model(RandomForestRegressor(n_estimators=n_estimators, random_state=42),
                          X_train, y_train,
                          X_test, y_test,
                          "Random Forest Regression")


def svr_regression():
    return evaluate_model(SVR(kernel="rbf"),
                          X_train_scaled, y_train,
                          X_test_scaled, y_test,
                          "Support Vector Regression (SVR)")


linear_regression()
polynomial_regression(degree=2)
decision_tree_regression()
random_forest_regression(n_estimators=100)
svr_regression()
```

```
def mean_squared_error(y_true, y_pred, *, sample_weight=None, mul
```

Open in tab   View source

Mean squared error regression loss.
Read more in the `User Guide <mean_squared_error>` .

Parameters

y_true : array-like of shape (n_samples,) or (n_samples, n_outputs)
  Ground truth (correct) target values.
y_pred : array-like of shape (n_samples,) or (n_samples, n_outputs)
  Estimated target values.
sample_weight : array-like of shape (n_samples,), default=None
  Sample weights

```
Linear Regression Performance:
R² Score: 0.9488
MAE: 1.1284
RMSE: 1.9075

Polynomial Regression (deg=2) Performance:
R² Score: -0.0591
MAE: 4.0771
RMSE: 8.6727

Decision Tree Regression Performance:
R² Score: 0.8959
MAE: 1.5058
RMSE: 2.7195

Random Forest Regression Performance:
R² Score: 0.9498
MAE: 1.1081
RMSE: 1.8876

Support Vector Regression (SVR) Performance:
R² Score: 0.8502
MAE: 1.9997
RMSE: 3.2615
```

▾ SVR ⓘ ?

```
SVR()
```

```
summary = pd.DataFrame(results, columns=["Model", "R² Score", "MAE", "RMSE"])
print(summary)
```

```
                           Model  R² Score       MAE      RMSE
0              Linear Regression  0.948769  1.128427  1.907490
1  Polynomial Regression (deg=2) -0.059051  4.077067  8.672741
2       Decision Tree Regression  0.895866  1.505758  2.719531
```

```
3         Random Forest Regression  0.949831  1.108082  1.887632
4  Support Vector Regression (SVR)  0.850226  1.999740  3.261493
```

Start coding or generate with AI.

Start coding or generate with AI.

```
def mean_squared_error(y_true, y_pred, *, sample_weight=None, mul
```

Open in tab View source

Mean squared error regression loss.
Read more in the `User Guide <mean_squared_error>` .

Parameters

y_true : array-like of shape (n_samples,) or (n_samples, n_outputs)
Ground truth (correct) target values.
y_pred : array-like of shape (n_samples,) or (n_samples, n_outputs)
Estimated target values.
sample_weight : array-like of shape (n_samples,), default=None
Sample weights