SVR on Epicurious dataset

Aim:

To implement SVR on the Epicurious dataset

Support Vector Regression:

To implement Support Vector Regression (SVR) on a synthetic 2D dataset that is non-linearly separable and analyze its performance.

Support Vector Regression (SVR) is an extension of Support Vector Machines (SVM) used for regression tasks. It uses kernel functions to handle non-linear relationships, fits the data within a margin, and controls error with a regularization parameter (C) to balance complexity and error tolerance.

SVR maps input features into a higher-dimensional space using kernels, fitting a regression line within a specified margin (epsilon). Points outside the margin are penalized based on the regularization parameter (C). The kernel choice (e.g., linear, polynomial, or RBF) helps capture complex relationships. After training, SVR predicts continuous values by transforming the learned function back to the original space.

```
import pandas as pd
import numpy as np

df=pd.read_csv('/content/epi_r.csv')

df
```


{"type":"dataframe","variable_name":"df"}


```
df.isna().sum()
```

```
title            0
rating           0
calories      4117
protein       4162
fat           4183
              ...
cookbooks        0
leftovers        0
snack            0
snack week       0
turkey           0
Length: 680, dtype: int64
```

```python
#df.dropna(subset=['calories','protein','fat','sodium'],inplace=True)
df=df.drop(columns=['title'])
df.fillna(df.median(), inplace=True)
```

```python
df.isnull().sum()
```

```
rating          0
calories        0
protein         0
fat             0
sodium          0
               ..
cookbooks       0
leftovers       0
snack           0
snack week      0
turkey          0
Length: 679, dtype: int64
```

```python
df
```

{"type":"dataframe","variable_name":"df"}

```python
def remove_outliers_iqr(df, threshold=1.5):
    for column in df.select_dtypes(include=[np.number]):
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - threshold * IQR
        upper_bound = Q3 + threshold * IQR

        df = df[(df[column] >= lower_bound) & (df[column] <=
upper_bound)]

    return df

df_cleaned = remove_outliers_iqr(df, threshold=1.0)

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.svm import SVR

y=df_cleaned['rating']
x=df_cleaned.drop(columns=['rating'])

scaler=StandardScaler()
x_scaled=scaler.fit_transform(x)

x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.
2,random_state=42)
```

```
svr=SVR(kernel='rbf',C=1,epsilon=0.1)
svr.fit(x_train,y_train)

y_pred=svr.predict(x_test)

r2=r2_score(y_test,y_pred)
print(f"R2 score: {r2}")
```

```
R2 score: 0.5204069873276727
```

```
svr = SVR()
param_grid = {
    "C": [0.1, 1, 10, 100],       # Regularization parameter
    "gamma": ["scale", "auto", 0.01, 0.1, 1],  # Kernel coefficient
    "epsilon": [0.01, 0.1, 0.2, 0.5]  # Margin of tolerance
}

# Perform Grid Search with 5-fold cross-validation
grid_search = GridSearchCV(svr, param_grid, scoring="r2", cv=5,
n_jobs=-1, verbose=1)
grid_search.fit(x_train, y_train.ravel())

# Get the best model
best_svr = grid_search.best_estimator_
```

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
```

```
<ipython-input-89-7f635c094406>:10: FutureWarning: Series.ravel is
deprecated. The underlying array is already 1D, so ravel is not
necessary.  Use `to_numpy()` for conversion to a numpy array instead.
  grid_search.fit(x_train, y_train.ravel())
```

```
# Predict on test data
y_pred = best_svr.predict(x_test)

# Evaluate performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print best hyperparameters and performance
print(f"Best Hyperparameters: {grid_search.best_params_}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"R² Score: {r2:.4f}")
```

```
Best Hyperparameters: {'C': 100, 'epsilon': 0.2, 'gamma': 'auto'}
Mean Squared Error (MSE): 0.0504
R² Score: 0.8123
```