```
import pandas as pd
```

```
df = pd.read_csv('/content/maaslar.csv')
```

```
X = df[['Egitim Seviyesi']]
y = df['maas']

print('Features (X) head:')
display(X.head())

print('\nTarget (y) head:')
display(y.head())
```

```
Features (X) head:
```

	Egitim Seviyesi
0	1
1	2
2	3
3	4
4	5

```
Target (y) head:
```

	maas
0	2250
1	2500
2	3000
3	4000
4	5500

```
dtype: int64
```

```
x_2d = X.values.reshape(-1, 1)
y_2d = y.values.reshape(-1, 1)

print('Shape of x_2d:', x_2d.shape)
print('Shape of y_2d:', y_2d.shape)
```

```
Shape of x_2d: (10, 1)
Shape of y_2d: (10, 1)
```

```
import numpy as np

X_array = X.values
y_array = y.values.reshape(-1, 1)

print('Shape of X_array:', X_array.shape)
print('Shape of y_array:', y_array.shape)
```

```
Shape of X_array: (10, 1)
Shape of y_array: (10, 1)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_array, y_array, test_size=0.3, random_state=42)

print('Shape of X_train:', X_train.shape)
print('Shape of X_test:', X_test.shape)
print('Shape of y_train:', y_train.shape)
print('Shape of y_test:', y_test.shape)
```

```
Shape of X_train: (7, 1)
Shape of X_test: (3, 1)
Shape of y_train: (7, 1)
Shape of y_test: (3, 1)
```

```
from sklearn.svm import SVR

# Initialize the SVR model with an RBF kernel
sup_vector_reg = SVR(kernel = 'rbf')
```

```
# Train the model
sup_vector_reg.fit(X_train, y_train.ravel())
```

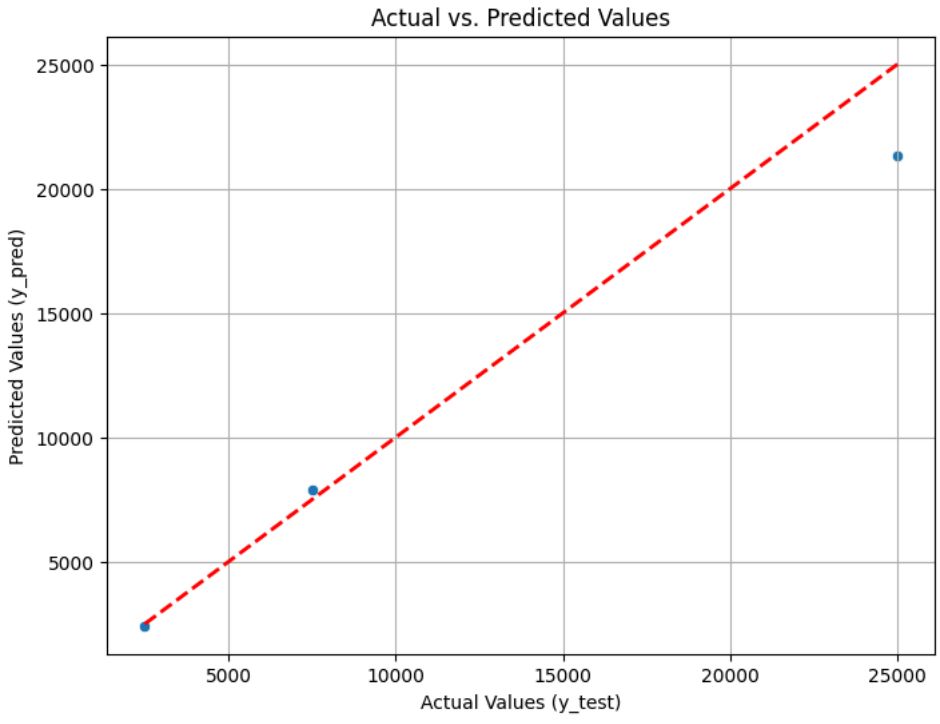
```
SVR
SVR()
```

```
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Get the best estimator from GridSearchCV
best_svr_model = grid_search.best_estimator_

# Make predictions on the test set
y_pred = best_svr_model.predict(X_test)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error (MSE): {mse:.2f}')

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error (RMSE): {rmse:.2f}')

# Calculate R-squared (R2)
r2 = r2_score(y_test, y_pred)
print(f'R-squared (R2): {r2:.2f}')
```

```
Mean Squared Error (MSE): 4538743.62
Root Mean Squared Error (RMSE): 2130.43
R-squared (R2): 0.95
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVR
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test.flatten(), y=y_pred.flatten())
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2) # Diagonal line
plt.xlabel('Actual Values (y_test)')
plt.ylabel('Predicted Values (y_pred)')
plt.title('Actual vs. Predicted Values')
plt.grid(True)
plt.show()
```




```
# Define the parameter grid
param_grid = {
    'kernel': ['rbf', 'poly', 'sigmoid'],
```

```
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1]
}

# Initialize GridSearchCV with reduced cv value
grid_search = GridSearchCV(SVR(), param_grid, cv=2, verbose=2, n_jobs=-1)

# Fit GridSearchCV
grid_search.fit(X_train, y_train.ravel())

# Print the best parameters and best score
print("Best parameters found: ", grid_search.best_params_)
print("Best score found: ", grid_search.best_score_)
```

```
Fitting 2 folds for each of 72 candidates, totalling 144 fits
Best parameters found:  {'C': 100, 'gamma': 'scale', 'kernel': 'poly'}
Best score found:  0.759488710396687
```

```
print("Optimal gamma value: ", grid_search.best_params_['gamma'])
```

```
Optimal gamma value:  scale
```

```
Start coding or generate with AI.
```