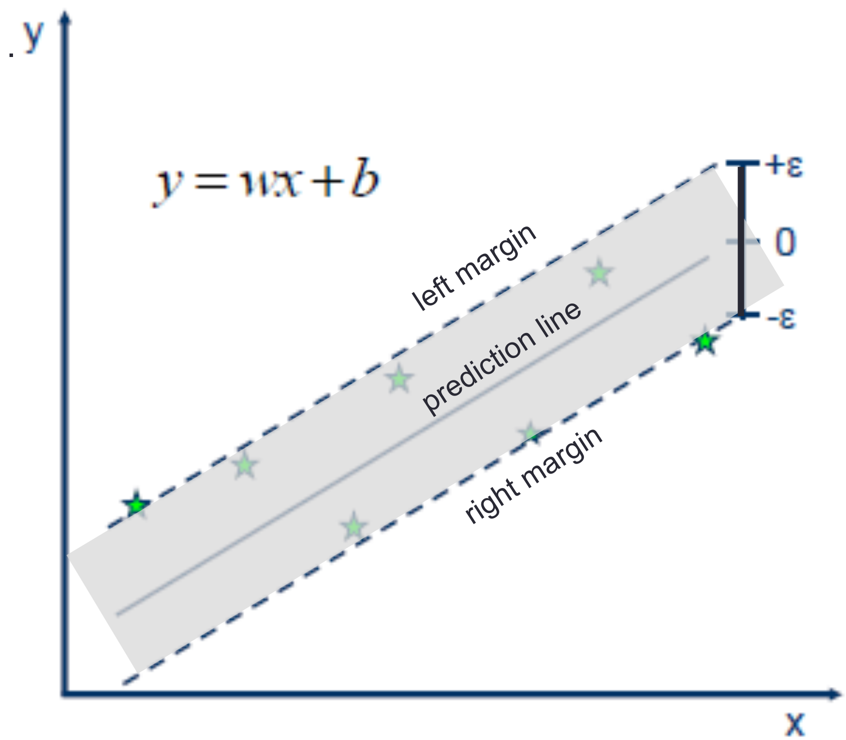
# SUPPORT VECTOR REGRESSION

# Outline

- Hard Margin Support Vector Regression
- Soft Margin Support Vector Regression
- Example 1 – Simple Support Vector Regression
- Example 2 – Multiple Support Vector Regression

# Hard Margin SVR




Solve for $b, w_1, \ldots, w_p$

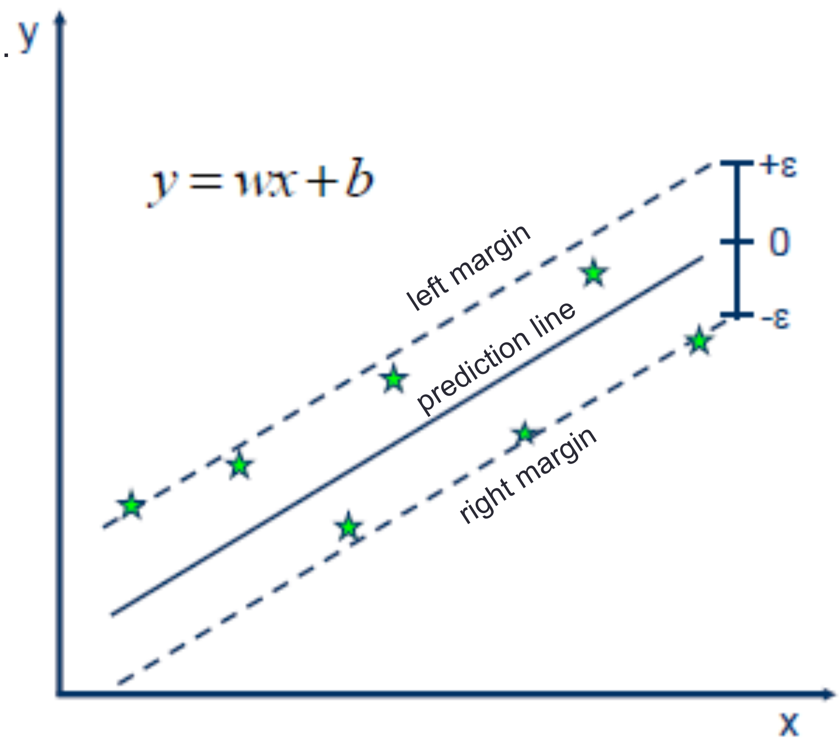
$$\text{Min} \quad \frac{1}{2}||\mathbf{w}||^2$$

$$\text{Subject to} \quad |y_i - \hat{y}_i| < \epsilon$$

$$i = 1, \ldots, n$$

- Want to find the smallest region (tube) around the fitted line, that includes all the data points
- The model has hyperparameter $\epsilon$
- For small $\epsilon$ values there is no feasible solution

# Hard Margin SVR




Solve for $b,\ w_1, \ldots, w_p$

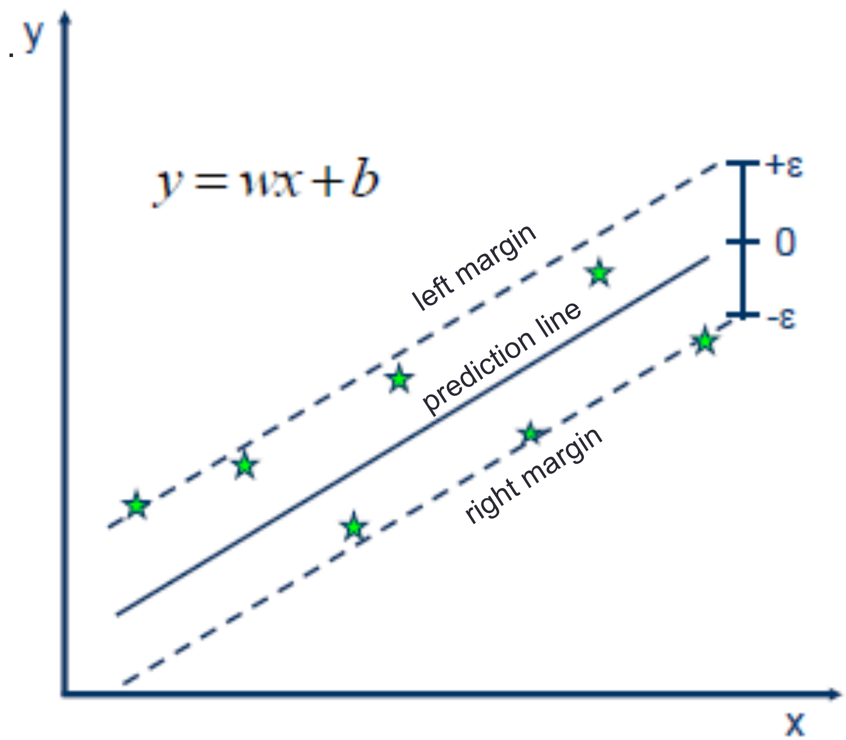
$$\text{Min} \quad \frac{1}{2}\left[w_1^2 + \cdots + w_p^2\right]$$

$$\text{Subject to} \quad |y_i - \hat{y}_i| < \epsilon$$

$$i = 1, \ldots, n$$

- Want to find the smallest region (tube) around the fitted line, that includes all the data points
- The model has hyperparameter $\epsilon$
- For small $\epsilon$ values there is no feasible solution

# Soft Margin SVR




- Allow data points to exceed (left or right) margins
- Maximize the number of data points between margins
- Only the cost of residuals larger than $\epsilon$ is accumulated in the loss function,
- Points within $\epsilon$-margins do not add cost and therefore have no effect on the regression equation

# SUPPORT VECTOR REGRESSION

Solve for $b,\ w_1, \ldots, w_p$

.

$$\text{Min} \qquad \frac{1}{2}\left[w_1^2 + \cdots + w_p^2\right] + C \sum_{i=1}^{n} L_i(b, w_1, \ldots, w_p)$$

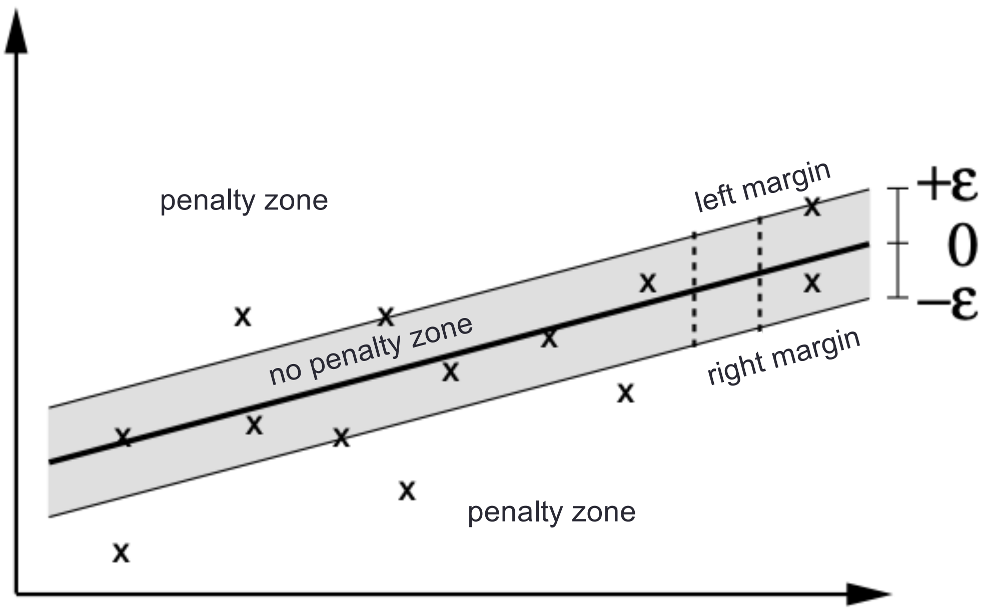
The $\epsilon$-insensitive loss function $L$ is

$$L_i(b, w_1, \ldots, w_p) = \begin{cases} 0 & \text{if} \quad |y_i - \hat{y}_i| < \epsilon \\ |y_i - \hat{y}_i| - \epsilon & \text{if} \quad |y_i - \hat{y}_i| > \epsilon \end{cases}$$

$$\text{where} \qquad \hat{y}_i = b + w_1\, x_{1i} + \cdots + w_p\, x_{pi}$$

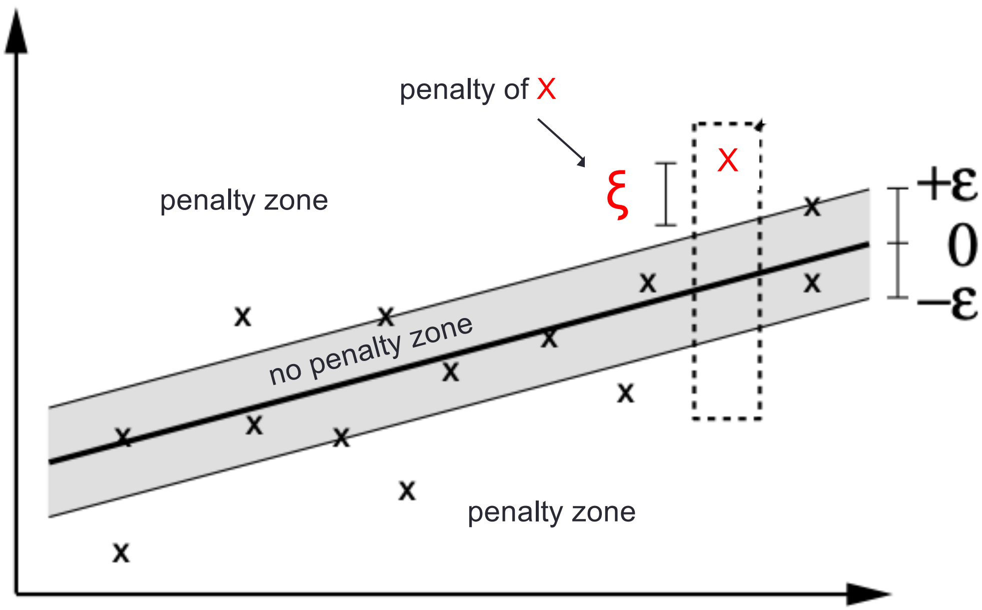
- Allow data points to exceed (left or right) margins
- Maximize the number of data points between margins
- Only the cost of residuals larger than $\epsilon$ is accumulated in the loss function,
- Points within $\epsilon$-margins do not add cost and therefore have no effect on the regression equation

# SUPPORT VECTOR REGRESSION



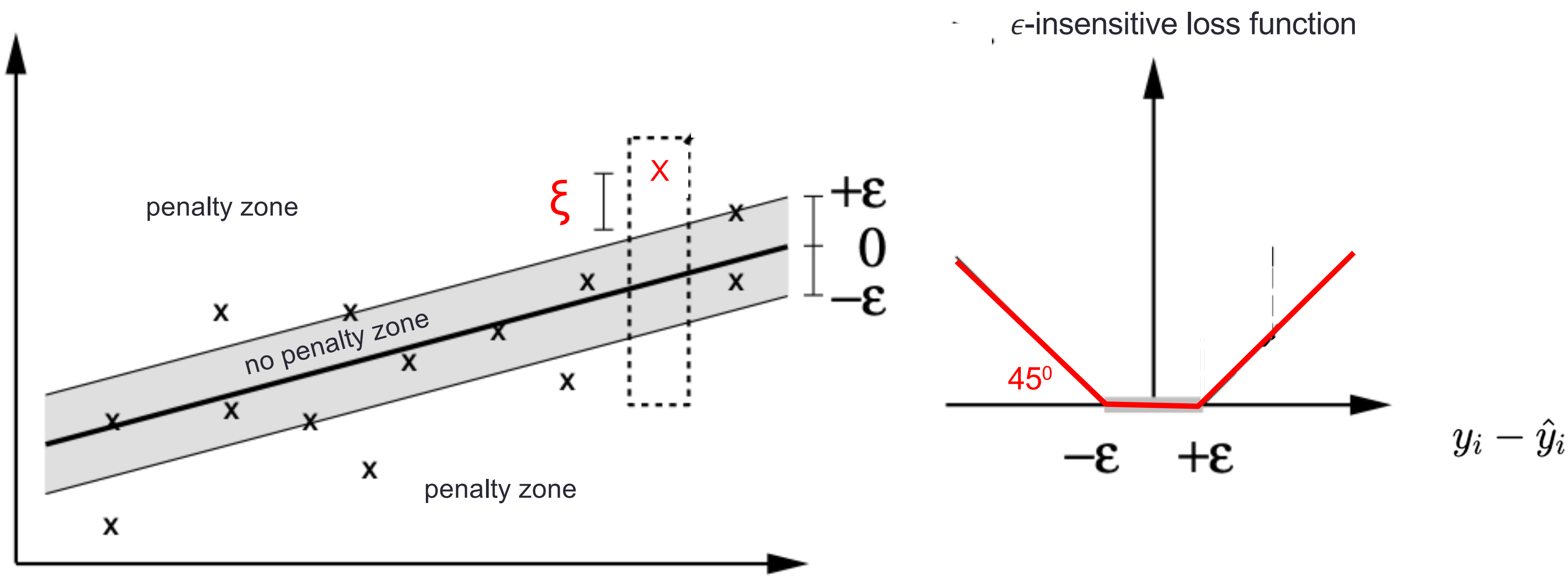

- A regression function is found such that a tube with radius ε around the regression function contains most data

# SUPPORT VECTOR REGRESSION



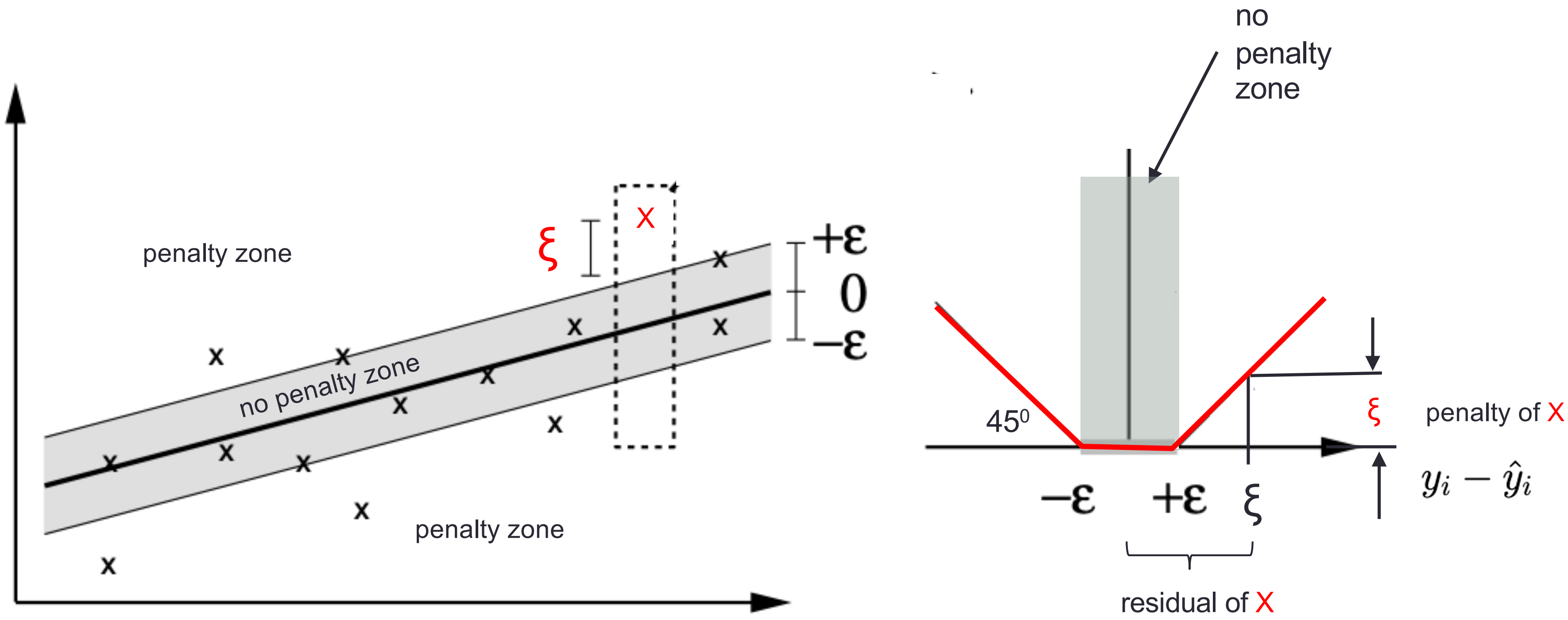

- A regression function is found such that a tube with radius ε around the regression function contains most data
- Points outside the tube are penalized

# SUPPORT VECTOR REGRESSION




- A regression function is found such that a tube with radius ε around the regression function contains most data
- Points outside the tube are penalized
- Penalty is given by the loss function

# SUPPORT VECTOR REGRESSION




- A regression function is found such that a tube with radius ε around the regression function contains most data
- Points outside the tube are penalized
- Penalty is given by the loss function

# EXAMPLE 1 – SVR

# Example 1 – SVR

**Predict the Price of used cars with Odometer readings**

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
```

```python
df1 = pd.read_csv('Odometer.csv')
```

| | Odometer | Price |
|---|---|---|
| **0** | 37.4 | 14.6 |
| **1** | 44.8 | 14.1 |
| **2** | 45.8 | 14.0 |
| **3** | 30.9 | 15.6 |
| **4** | 31.7 | 15.6 |
| **...** | ... | ... |
| **95** | 36.2 | 14.8 |
| **96** | 34.2 | 14.6 |
| **97** | 33.2 | 14.5 |
| **98** | 39.2 | 14.7 |
| **99** | 36.4 | 14.3 |

100 rows × 3 columns

# Example 1 – SVR

### Predict the Price of used cars with Odometer readings

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
```

```
df1 = pd.read_csv('Odometer.csv')
```

```
Price = df1.Price
Odometer = df1.drop(["Price"],axis=1)
```

### Fit Linear Regression model

```
m1 = LinearRegression()
m1.fit(Odometer,Price)
m1.score(Odometer,Price)
```

```
0.6482954749384247
```

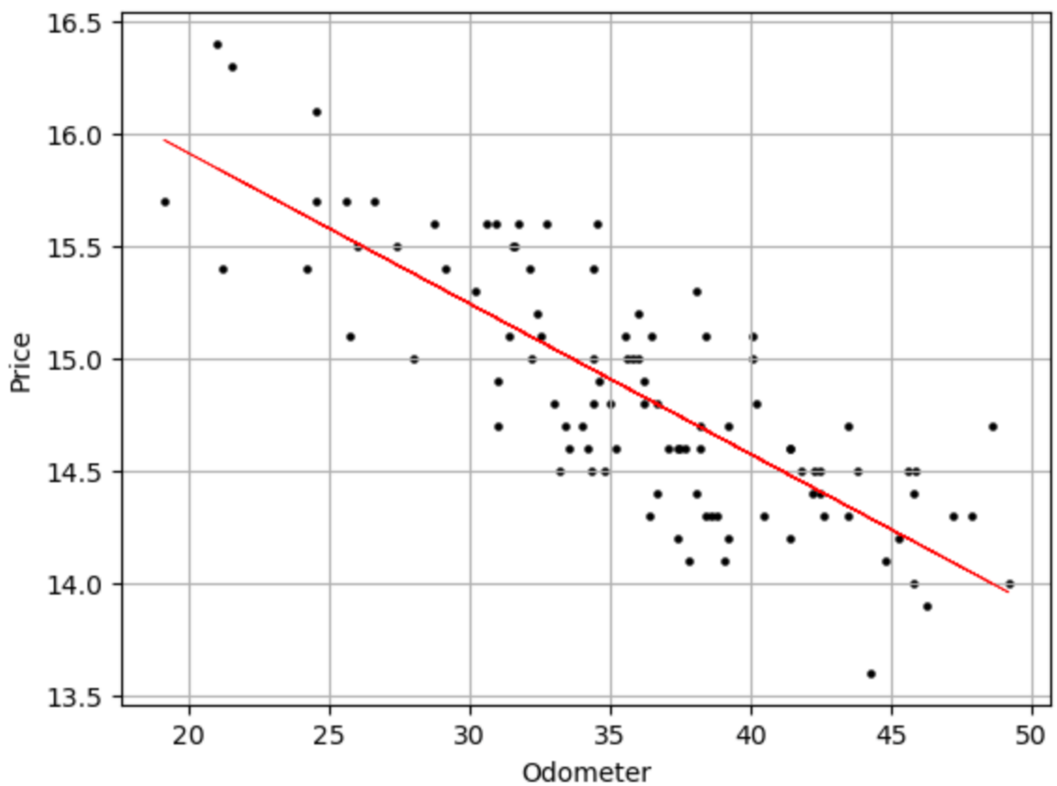
```
yhat = m1.predict(Odometer)
df2 = df1.copy()
df2['prediction'] = yhat
```

| | Odometer | Price | prediction |
|---|---|---|---|
| **0** | 37.4 | 14.6 | 14.748130 |
| **1** | 44.8 | 14.1 | 14.253360 |
| **2** | 45.8 | 14.0 | 14.186499 |
| **3** | 30.9 | 15.6 | 15.182726 |
| **4** | 31.7 | 15.6 | 15.129237 |
| **...** | ... | ... | ... |
| **95** | 36.2 | 14.8 | 14.828363 |
| **96** | 34.2 | 14.6 | 14.962085 |
| **97** | 33.2 | 14.5 | 15.028946 |
| **98** | 39.2 | 14.7 | 14.627781 |
| **99** | 36.4 | 14.3 | 14.814991 |

100 rows × 3 columns

# Example 1 – SVR

```
plt.figure()
plt.scatter(Odometer,Price,c='k',s=5)
plt.plot(Odometer,yhat,color = 'r',linewidth = 0.5)
plt.ylabel('Price')
plt.xlabel('Odometer')
```




```
yhat = m1.predict(Odometer)
df2 = df1.copy()
df2['prediction'] = yhat
```

| | Odometer | Price | prediction |
|---|---|---|---|
| **0** | 37.4 | 14.6 | 14.748130 |
| **1** | 44.8 | 14.1 | 14.253360 |
| **2** | 45.8 | 14.0 | 14.186499 |
| **3** | 30.9 | 15.6 | 15.182726 |
| **4** | 31.7 | 15.6 | 15.129237 |
| **...** | ... | ... | ... |
| **95** | 36.2 | 14.8 | 14.828363 |
| **96** | 34.2 | 14.6 | 14.962085 |
| **97** | 33.2 | 14.5 | 15.028946 |
| **98** | 39.2 | 14.7 | 14.627781 |
| **99** | 36.4 | 14.3 | 14.814991 |

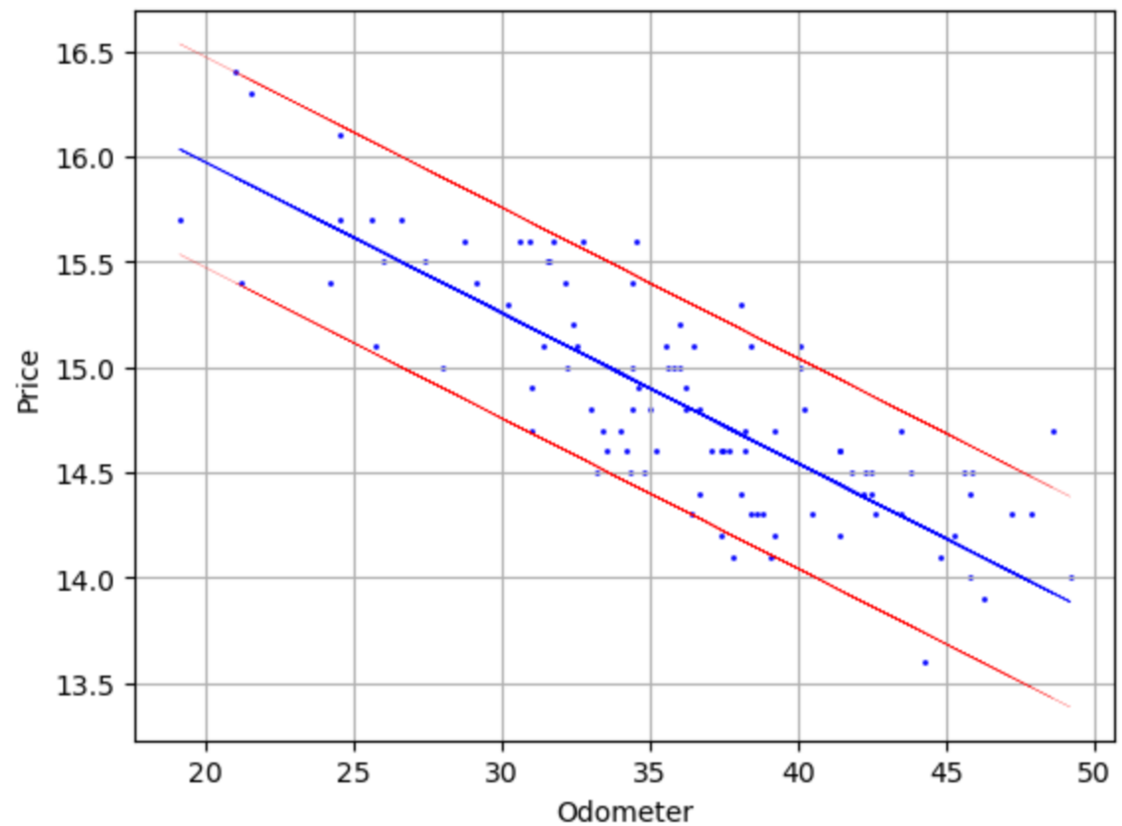
100 rows × 3 columns

# Example 1 – SVR

## Build the SVR Model

```
epsilon2 = 0.5
svr2 = SVR(kernel='linear',C=1.0,
           epsilon = epsilon2)
svr2.fit(Odometer, Price);
yhat2 = svr2.predict(Odometer)
df2['prediction2'] = yhat2
left2 = yhat2 + epsilon2
right2 = yhat2 - epsilon2
```

```
plt.figure()
plt.scatter(Odometer,Price,
            c='b',s=1)
plt.plot(Odometer,yhat2,
         color = 'b',
         linewidth = 0.5)
plt.plot(Odometer,left2,
         color = 'r',
         linewidth = 0.1)
plt.plot(Odometer,right2,
         color = 'r',
         linewidth = 0.1)
plt.ylabel('Price')
plt.xlabel('Odometer')
```

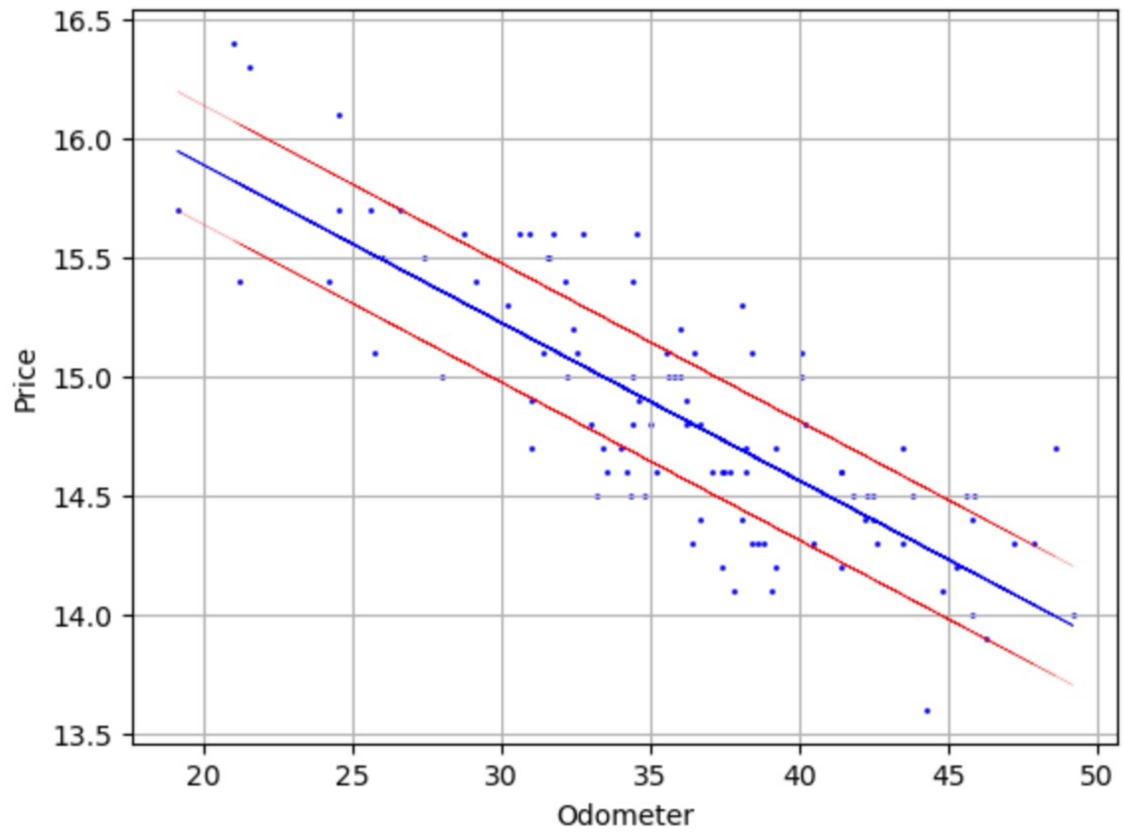


# Example 1 – SVR

## Build the SVR Model

```
epsilon2 = 0.25
svr2 = SVR(kernel='linear',C=1.0,
           epsilon = epsilon2)
svr2.fit(Odometer, Price);
yhat2 = svr2.predict(Odometer)
df2['prediction2'] = yhat2
left2 = yhat2 + epsilon2
right2 = yhat2 - epsilon2
```

```
plt.figure()
plt.scatter(Odometer,Price,
            c='b',s=1)
plt.plot(Odometer,yhat2,
         color = 'b',
         linewidth = 0.5)
plt.plot(Odometer,left2,
         color = 'r',
         linewidth = 0.1)
plt.plot(Odometer,right2,
         color = 'r',
         linewidth = 0.1)
plt.ylabel('Price')
plt.xlabel('Odometer')
```

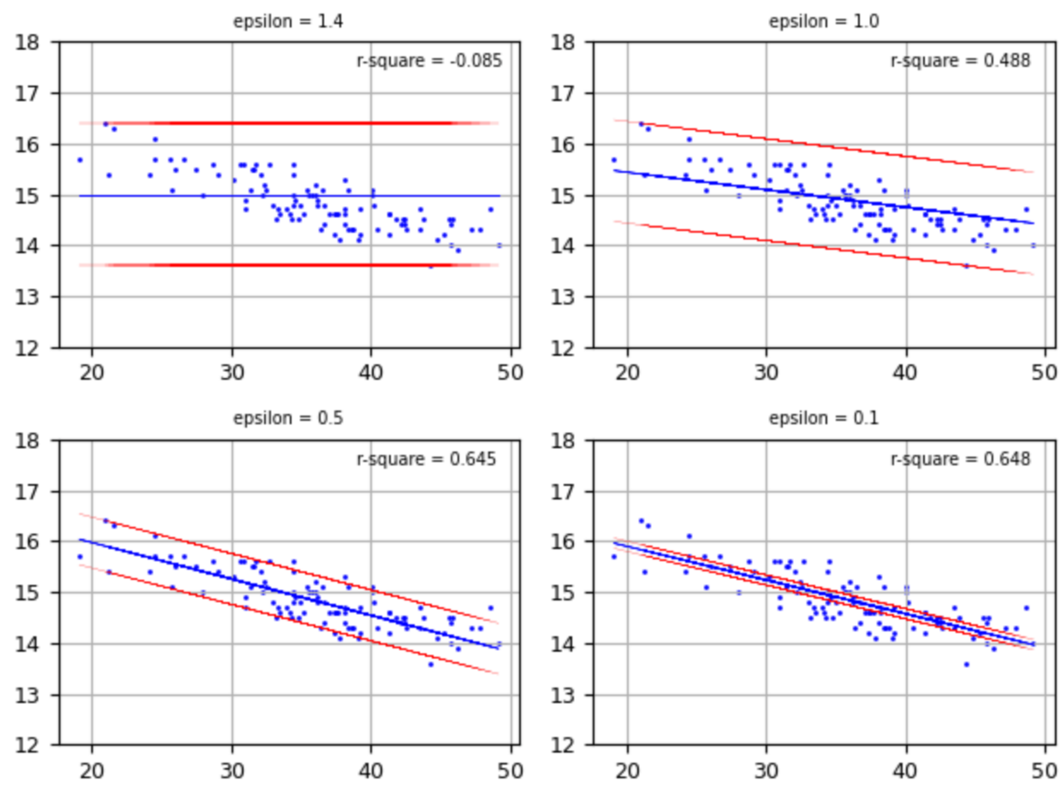


# Example 1 – SVR

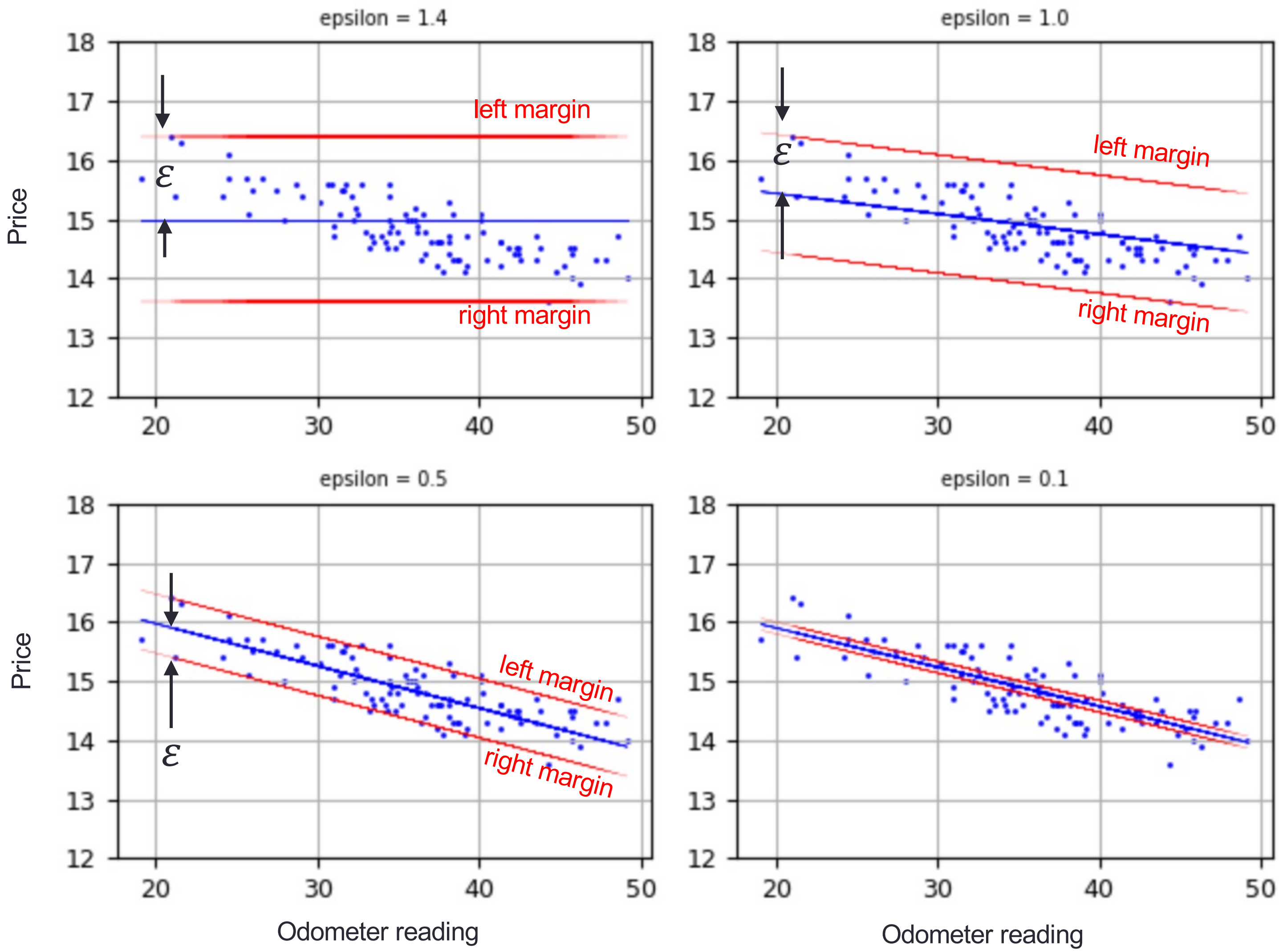
```
eps = [1.4,1.0,0.5,0.1]

plt.figure()
for j in range(4):
    svr = SVR(kernel='linear',C=1.0,
              epsilon = eps[j])
    svr.fit(Odometer, Price);
    yhat = svr.predict(Odometer)
    left = yhat + eps[j]
    right = yhat - eps[j]


    plt.subplot(2,2,j+1)
    plt.scatter(Odometer,Price,c='b',s=1)
    plt.plot(Odometer,yhat,color = 'b',
             linewidth = 0.5)
    plt.plot(Odometer,left,color = 'r',
             linewidth = 0.1)
    plt.plot(Odometer,right,color = 'r',
             linewidth = 0.1)
    r2=round(svr.score(Odometer,Price),3)
    plt.annotate('r-square = {}'.format(r2),
                 (39,17.5),fontsize=7)
    plt.title('epsilon = {}'.format(eps[j]),
              fontsize=7)
```



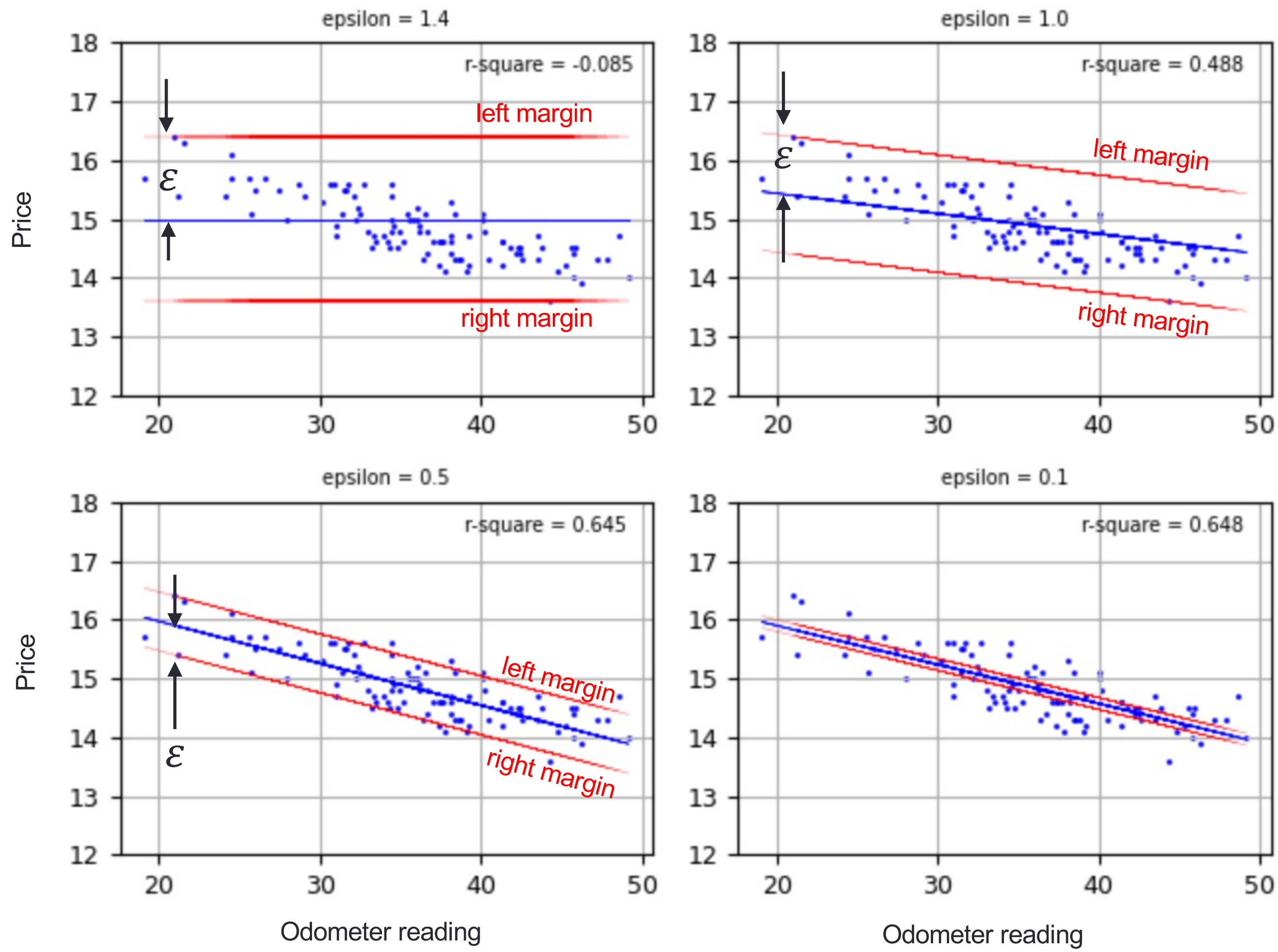
# SUPPORT VECTOR REGRESSION




$C$ = 1 in all cases

SVR fits the flattest possible linear function trying to capture most data points

# SUPPORT VECTOR REGRESSION




$C$ = 1 in all cases

SVR fits the flattest possible linear function trying to capture most data points

# EXAMPLE 2 – SVR

# Example 2 – SVR

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

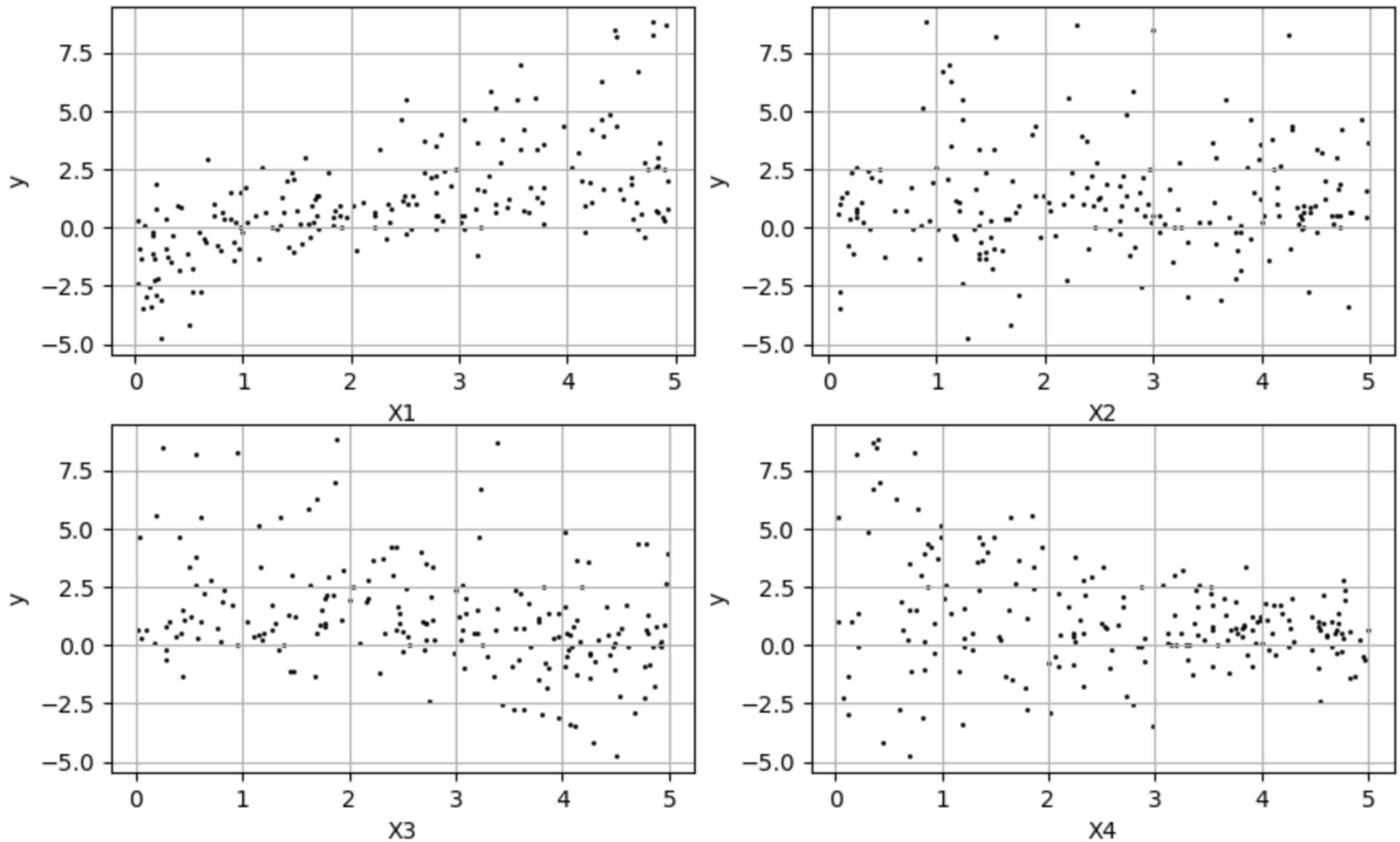
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
```

```
df = pd.read_csv('svr.csv')
df.head()
```

| | X1 | X2 | X3 | X4 | y |
|---|---|---|---|---|---|
| **0** | 4.480614 | 1.363373 | 4.404379 | 3.399033 | 1.651549 |
| **1** | 0.149057 | 4.808124 | 4.067836 | 1.200827 | -3.389763 |
| **2** | 4.162641 | 1.765042 | 1.313687 | 4.577033 | 0.977445 |
| **3** | 1.266521 | 3.202102 | 1.386985 | 3.580701 | 0.027955 |
| **4** | 3.171857 | 4.987052 | 2.216417 | 1.386676 | 3.632987 |

# Example 2 – SVR



# Example 2 – SVR

```
X = df[['X1','X2','X3','X4']]
y = df['y']
```

```
X_train, X_test, y_train, y_test =\
        train_test_split(X, y,
                         test_size=0.33,
                         random_state=42)
```

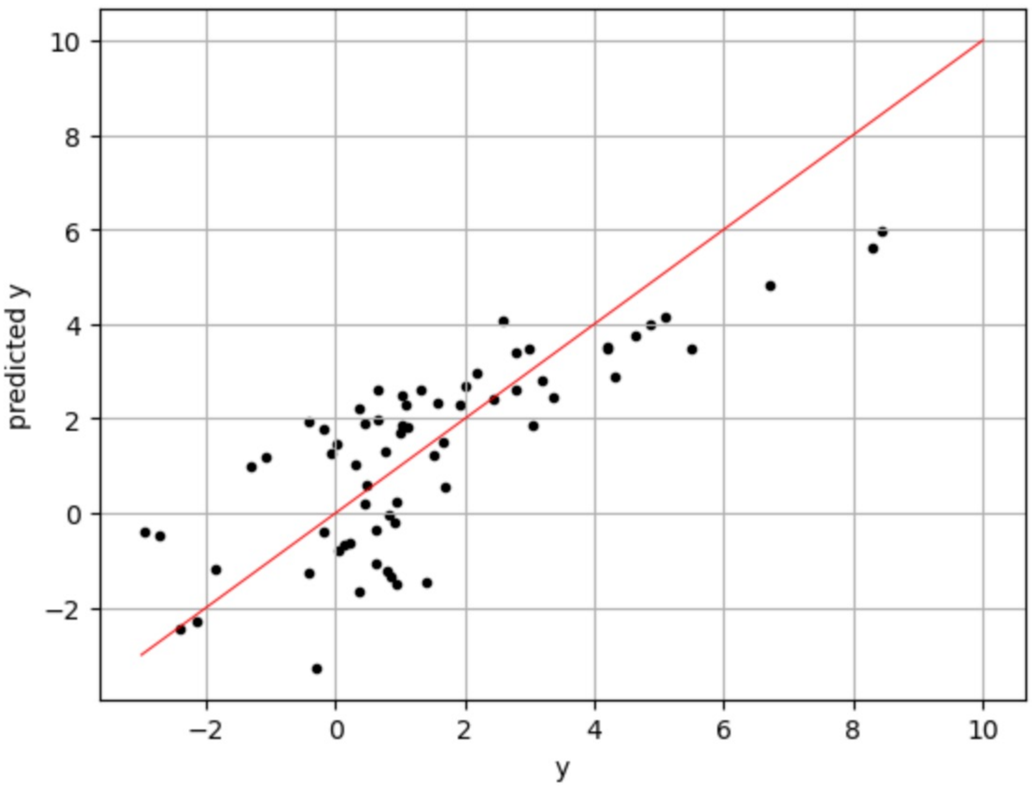
**Linear regression as baseline**

```
linear = LinearRegression()
linear.fit(X_train,y_train)
linear.score(X_test,y_test)
```

```
0.6086552006495394
```

r-squared

```
yhat = linear.predict(X_test)
plt.scatter(y_test,yhat,s=9,color='k')
plt.plot(xaxis,yaxis,color='r',linewidth=0.7)
plt.xlabel('y')
plt.ylabel('predicted y')
```



# Example 2 – SVR

### Support vector regressor with linear kernel

```
svr_linear = SVR(kernel='linear')
svr_linear.fit(X_train, y_train)
svr_linear.score(X_test,y_test)
```

```
0.6303795162411019
```

```
svr_linear = SVR(kernel='linear',
                 gamma='scale', C=1.0,
                 epsilon=0.1)
svr_linear.fit(X_train, y_train)
svr_linear.score(X_test,y_test)
```

```
0.6303795162411019
```

r-squared

### Support vector regressor with RBF kernel

```
svr_rbf = SVR(kernel='rbf')
svr_rbf.fit(X_train, y_train);
svr_rbf.score(X_test,y_test)
```

```
0.7304232109135085
```

r-squared

# Example 2 – SVR

Kernel types

- linear
- polynomial
- radial basis function (RBF)
- sigmoid

# Example 2 – SVR

### Support vector regressor with linear kernel

```
svr_linear = SVR(kernel='linear')
svr_linear.fit(X_train, y_train)
svr_linear.score(X_test,y_test)
```

```
0.6303795162411019
```

```
svr_linear = SVR(kernel='linear',
                 gamma='scale', C=1.0,
                 epsilon=0.1)
svr_linear.fit(X_train, y_train)
svr_linear.score(X_test,y_test)
```

```
0.6303795162411019
```

r-squared

### Support vector regressor with RBF kernel

```
svr_rbf = SVR(kernel='rbf')
svr_rbf.fit(X_train, y_train);
svr_rbf.score(X_test,y_test)
```

```
0.7304232109135085
```

r-squared

square-root of MSPE

```
print("RMSE for linear SVR:",
      np.sqrt(mean_squared_error(y_test,yhat0)))
print("RMSE for RBF kernelized SVR:",
      np.sqrt(mean_squared_error(y_test,yhat)))
```

```
RMSE for linear SVR: 1.3874704298641074
RMSE for RBF kernelized SVR: 1.1849143760665064
```

# GridSearchCV - Find best C and $\varepsilon$

```
params = {'C':[0.01,0.05,0.1,0.5,1,2,5],
          'epsilon':[0.1,0.2,0.5,1]}
```

```
grid = GridSearchCV(svr_rbf,param_grid=params,
                    cv=5,scoring='r2')
grid.fit(X_train,y_train);
```

```
grid.best_params_
```

```
{'C': 5, 'epsilon': 0.5}
```

```
svr_best = grid.best_estimator_
svr_best.fit(X_train, y_train)
svr_best.score(X_test,y_test)
```

```
0.7862999176421599
```

# GridSearchCV - Find best C and $\varepsilon$

```
params = {'C':[0.01,0.05,0.1,0.5,1,2,5],
          'epsilon':[0.1,0.2,0.5,1]}
```

```
grid = GridSearchCV(svr_rbf,param_grid=params,
                    cv=5,scoring='r2')
grid.fit(X_train,y_train);
```

```
grid.best_params_
```

```
{'C': 5, 'epsilon': 0.5}
```

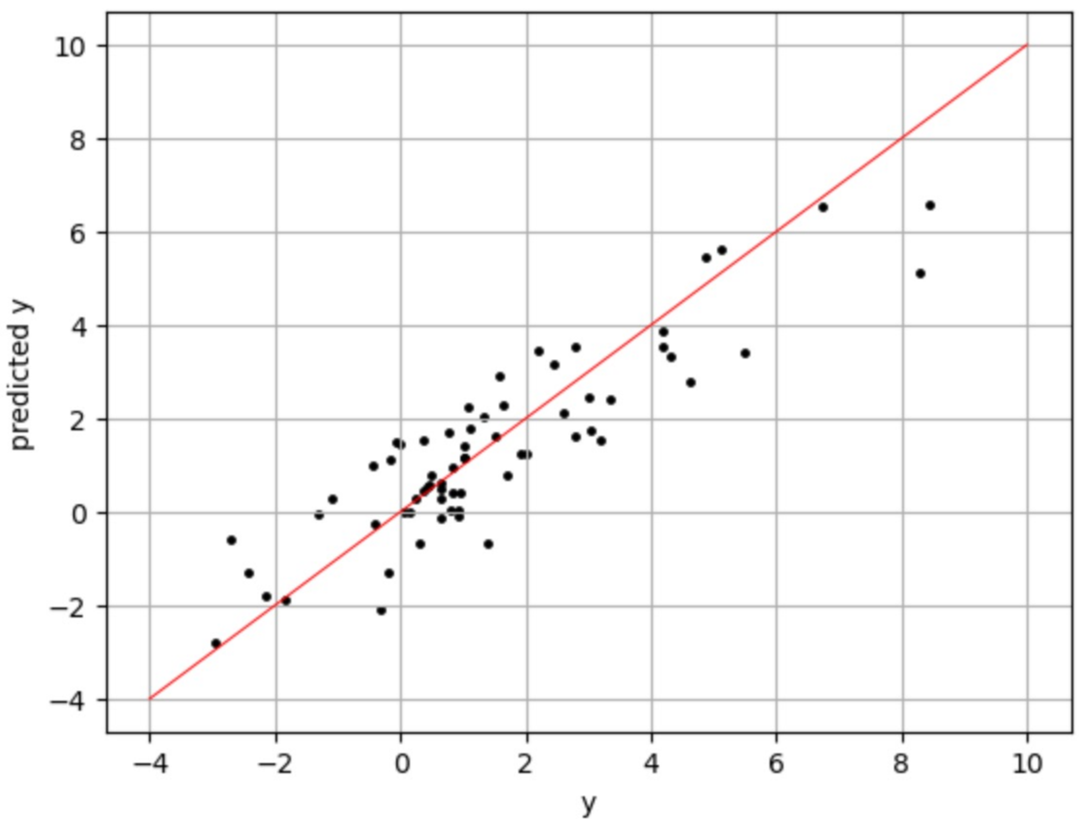
```
svr_best = grid.best_estimator_
svr_best.fit(X_train, y_train)
svr_best.score(X_test,y_test)
```

```
0.7862999176421599
```

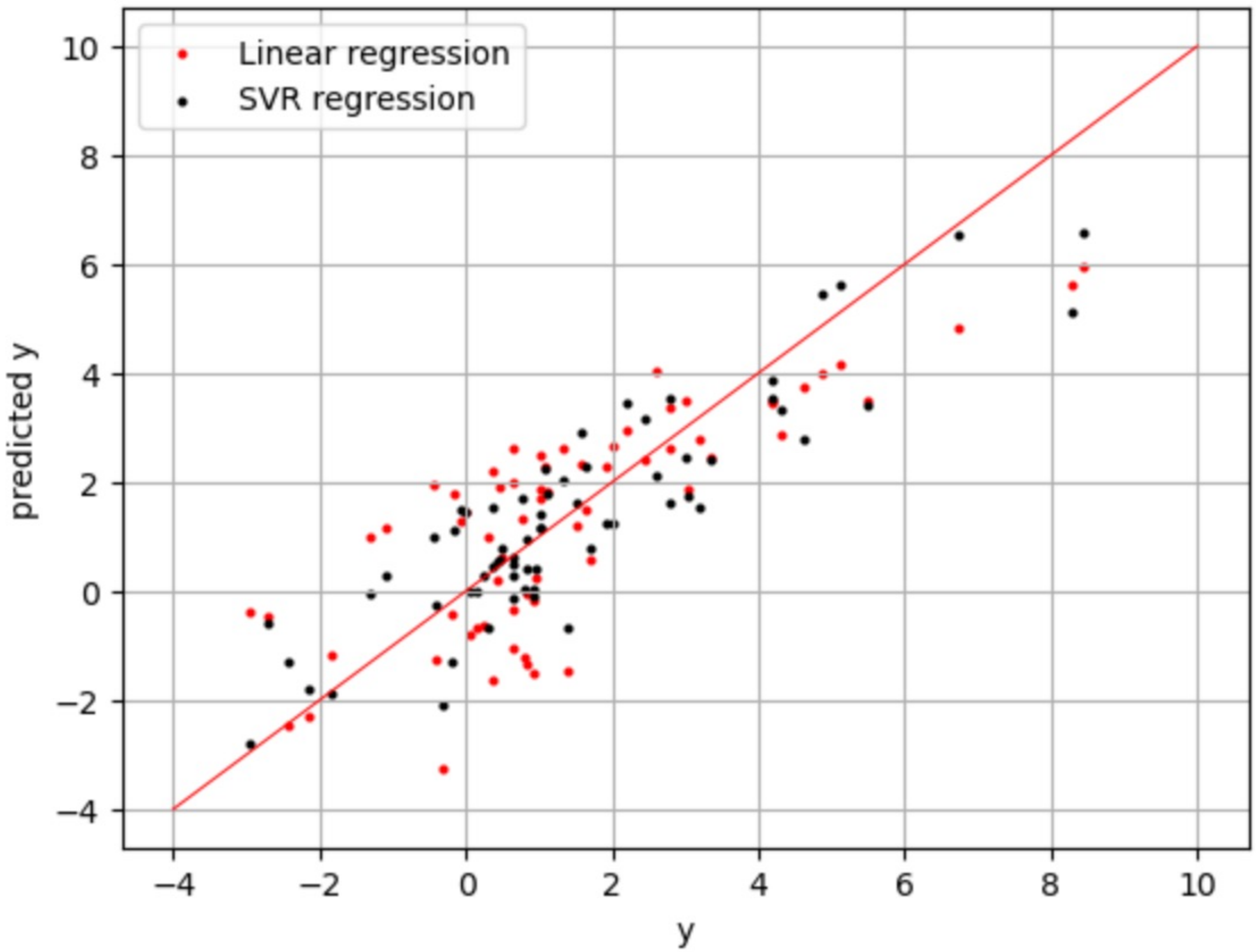
```
yhat = svr_best.predict(X_test)

print("RMSE for RBF kernelized SVR:",
      np.sqrt(mean_squared_error(y_test,yhat)))
```

```
RMSE for RBF kernelized SVR: 1.054989325250114б
```



# Example 2 – SVR vs Linear Regression



# Example 2 – SVR vs Linear Regression