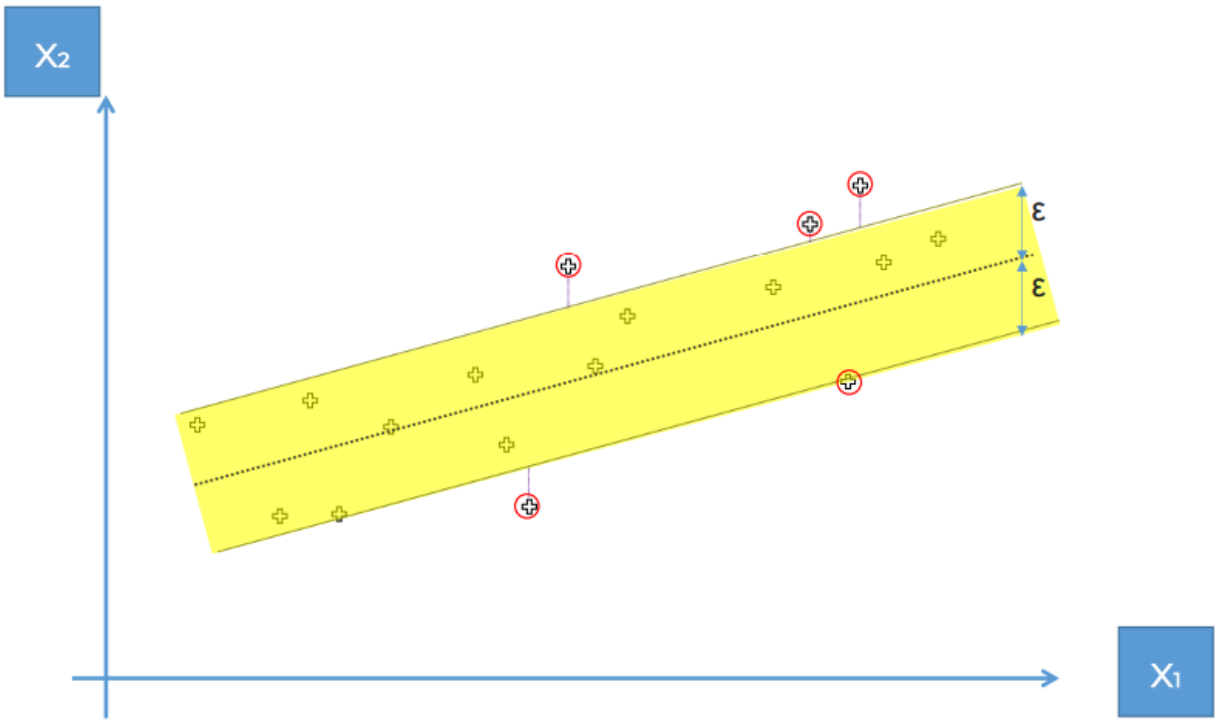
# Support Vector Regression

## Understanding

- In a linear regression, we find the trendline using the Ordinary Least Squares method, essentially a line with a minimum error for all data points. But in the Support Vector Regression, we have a tube called $\epsilon$ - insensitive tube with a width from the trendline (located in the middle), this width is $\epsilon$.
- In the bellow diagram, data points part of the tube or within the width from trendline, $\epsilon$, have their error's disregarded. Meaning any distance between data point and trendline is disregarded.
- But the points outside have their errors taken into measurement. The points bellow the tube are called $\xi_n^*$ and the data points above the tube are called $\xi_n$. We measure their data point to the $\epsilon$-insensitive tube, not the middle trendline.




- Using the equation:

$$1/2||w||^2 + C\sum_{i}^{m}(\xi_i^* + \xi_i) -> min$$

- Essentially highlighting the fact that the sum of $\xi_i^* + \xi_i$ should be minimum.

## Why Called Support Vector Regression?

- Essentially all of these data points outside of the tube, are essentially vectors. But these outside points dictate the shape or formation of the tube, hence called support vectors.

## Example:

We are using the same example referenced in the Polynomial Regression, where we have a non-linear relation between the matrix of features `Level`, relating to the position, and the dependent matrix of feature, the `salary`.

## Applying Support Vector Regressions

- **Import Libraries**

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

- **Import Dataset**

```
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:-1].values
y = dataset.iloc[:, -1].values
```

- **Apply Feature Scaling**

```
y = y.reshape(len(y),1)
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

First step is to convert the y to a 2-D array. This is possible with the help of the reshape function.
Next we import the `StandardScaler` from `sklearn.preprocessing`, and create two instances of it, `sc_X` and `sc_Y`. We need two instances as this time, we need to apply feature scaling to both dependent column and the matrix of features.
Simply apply `fit_transform` to both `X,y`.

- **Training the SVR model on the whole dataset**

```
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X, y)
```

Import the `SVR` class from our `scikit` library. Create an instance called `regressor` and call the `SVR`. We have something called kernels, which can learn linear linear relationships in the datasets, or non-linear relations called `rbf`. We have a non-linear relation, hence use `rbf` also known as Gaussian Radial Basis Function.
Simply train the model, by using the `fit` function.

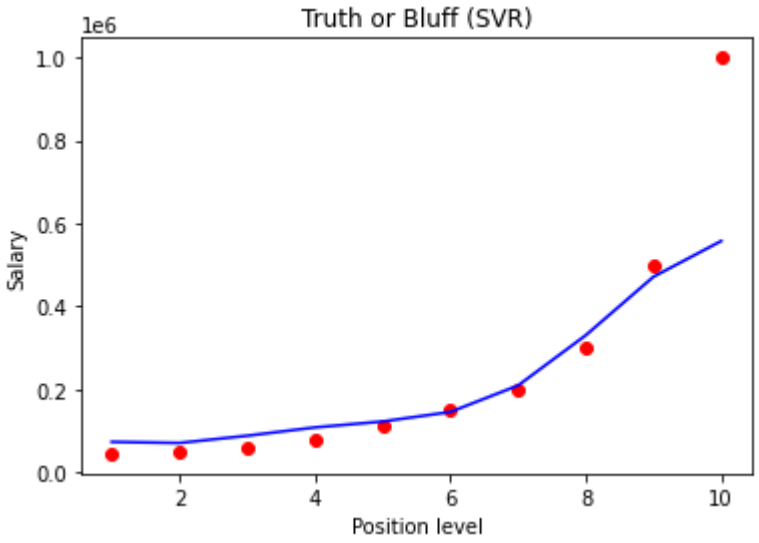
- **Predicting a new result**

```
sc_y.inverse_transform(regressor.predict(sc_X.transform([[6.5]]
)).reshape(-1,1))
```

Because we applied feature scaling on our results using the `StandardScaler`, our results will be scaled. Hence we need to perform the inverse.
Using the `sc_y` and use the `inverse_transformation` we can revert the prediction from scaled value to actual value.
To predict, use the regressor and `predict` method and pass the transformed value of whatever you want to predict, in this case, level = 6.5.
To prevent format errors, apply the reshape.

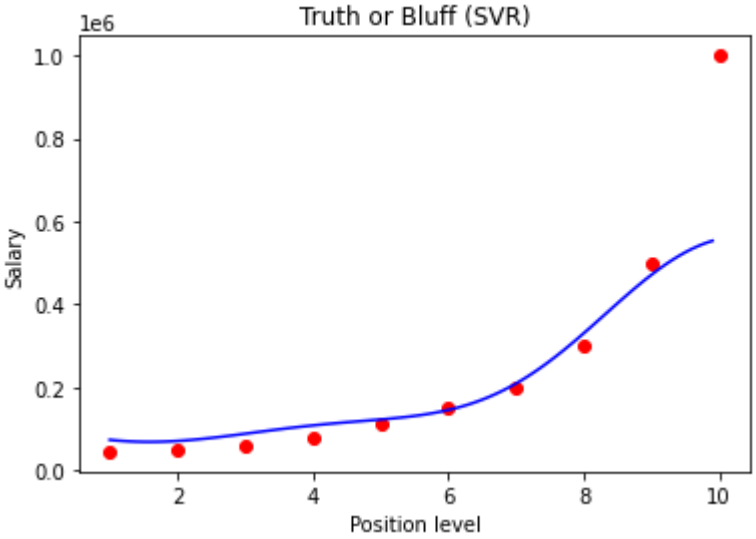
- **Visualizing the SVR results**

```
plt.scatter(sc_X.inverse_transform(X),
sc_y.inverse_transform(y), color = 'red')
plt.plot(sc_X.inverse_transform(X),
sc_y.inverse_transform(regressor.predict(X).reshape(-1,1)),
color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```




To make the plot for higher resolution and smother curves:

```
X_grid = np.arange(min(sc_X.inverse_transform(X)),
max(sc_X.inverse_transform(X)), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(sc_X.inverse_transform(X),
sc_y.inverse_transform(y), color = 'red')
plt.plot(X_grid,
sc_y.inverse_transform(regressor.predict(sc_X.transform(X_grid)
).reshape(-1,1)), color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()**
```


Truth or Bluff (SVR)
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Salary
2
4
6
8
10
Position level