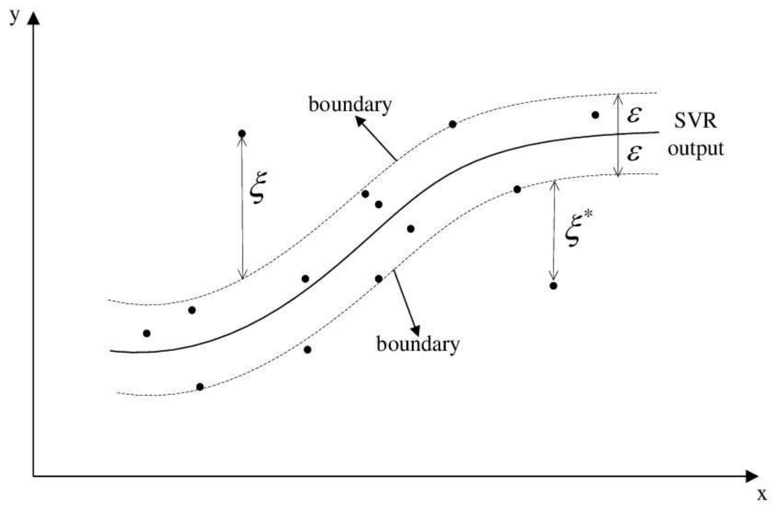
SUPPORT VECTOR REGRESSION (SVR)

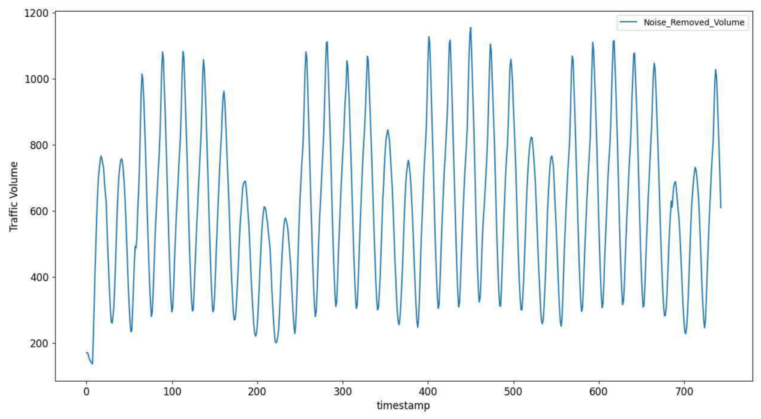
Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed.
Support vector regression is a supervised learning algorithm used to predict discrete values. Supports vector regression uses the same principle as SVM. The basic idea behind SVR is to find the most suitable line.
In SVR, the best fit line is the hyperplane with the maximum number of points.



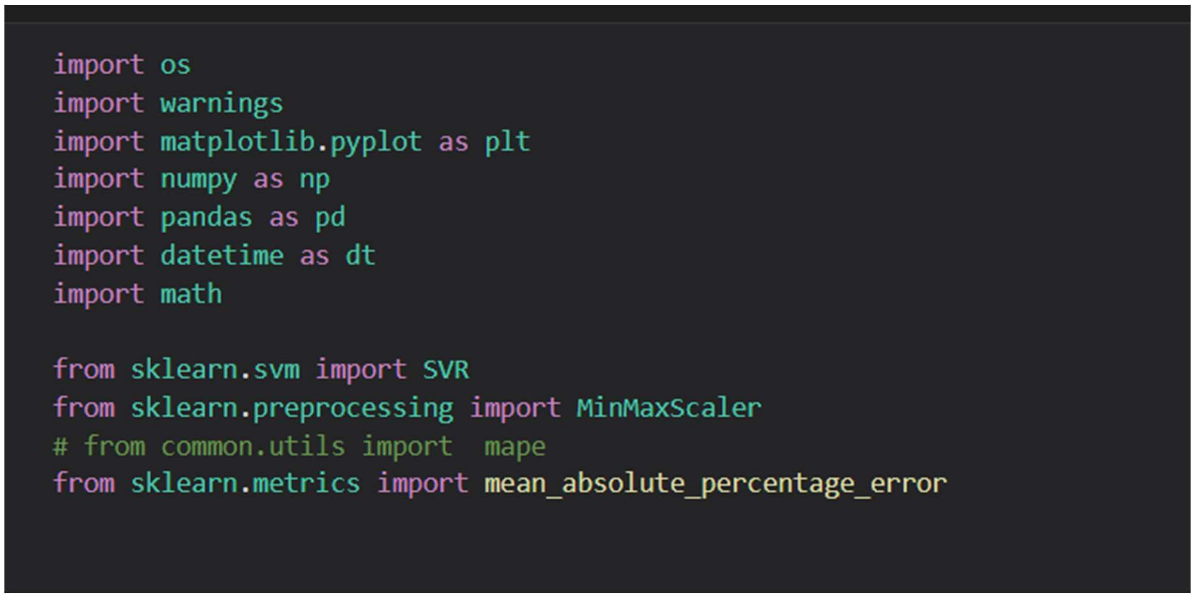
Source:

- Allows to choose error tolerance (**ϵ**)
- Tolerance for values falling outside acceptable error frame (**C**) or ξ^*

After successfully loading the data, the next step is to visualize this data.



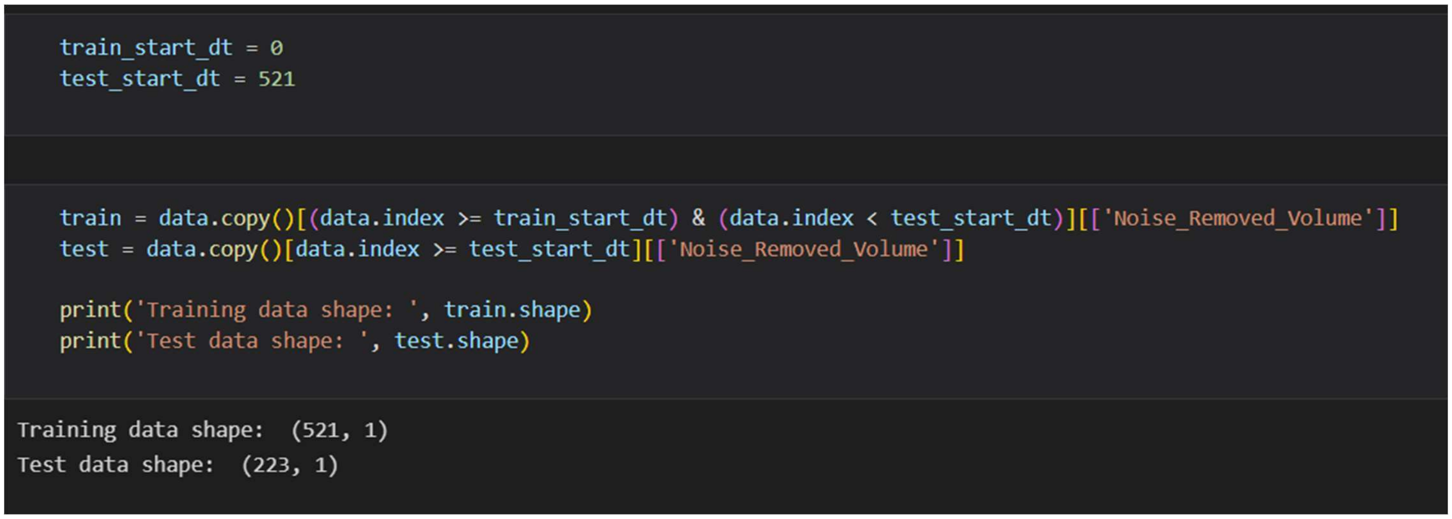
The libraries used and defined are:



```
import os
import warnings
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
import math

from sklearn.svm import SVR
from sklearn.preprocessing import MinMaxScaler
# from common.utils import  mape
from sklearn.metrics import mean_absolute_percentage_error
```

The data is divided into 70% training set and 30% test set as shown below.



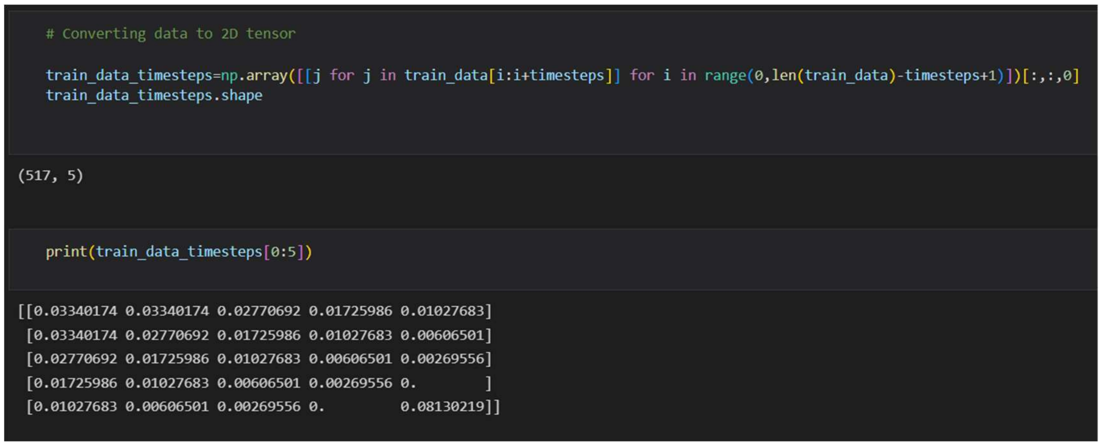
```
train_start_dt = 0
test_start_dt = 521
```

```
train = data.copy()[(data.index >= train_start_dt) & (data.index < test_start_dt)][['Noise_Removed_Volume']]
test = data.copy()[data.index >= test_start_dt][['Noise_Removed_Volume']]

print('Training data shape: ', train.shape)
print('Test data shape: ', test.shape)
```

```
Training data shape:  (521, 1)
Test data shape:  (223, 1)
```

Following, the train-test sets is divided into 2D tensor where for every 4 inputs, there is 1 output.



```
# Converting data to 2D tensor

train_data_timesteps=np.array([[j for j in train_data[i:i+timesteps]] for i in range(0,len(train_data)-timesteps+1)])[:,:,0]
train_data_timesteps.shape
```

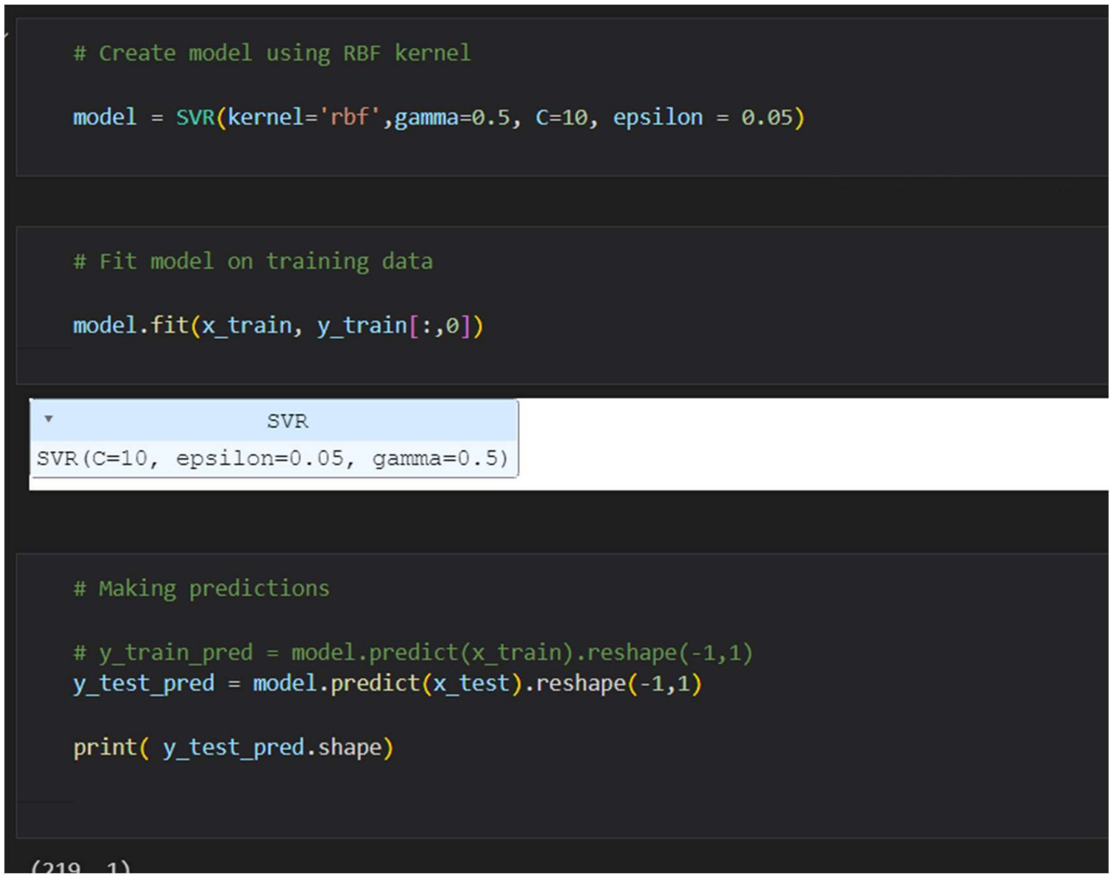
```
(517, 5)
```

```
print(train_data_timesteps[0:5])
```

```
[[0.03340174 0.03340174 0.02770692 0.01725986 0.01027683]
 [0.03340174 0.02770692 0.01725986 0.01027683 0.00606501]
 [0.02770692 0.01725986 0.01027683 0.00606501 0.00269556]
 [0.01725986 0.01027683 0.00606501 0.00269556 0.        ]
 [0.01027683 0.00606501 0.00269556 0.         0.08130219]]
```

SVR() function is used from "sklearn.svm" pckage to define the regression based model. In hyperparameters

- Kernel defines the transformation function (non-linear in this case)
- Epsilon defines error tolerance (ϵ)
- 'C' defines Tolerance for values falling outside acceptable error frame
- Gamma is Kernel coefficient and defines the reach of kernel



```
# Create model using RBF kernel

model = SVR(kernel='rbf',gamma=0.5, C=10, epsilon = 0.05)
```

```
# Fit model on training data

model.fit(x_train, y_train[:,0])
```

```
SVR
SVR(C=10, epsilon=0.05, gamma=0.5)
```

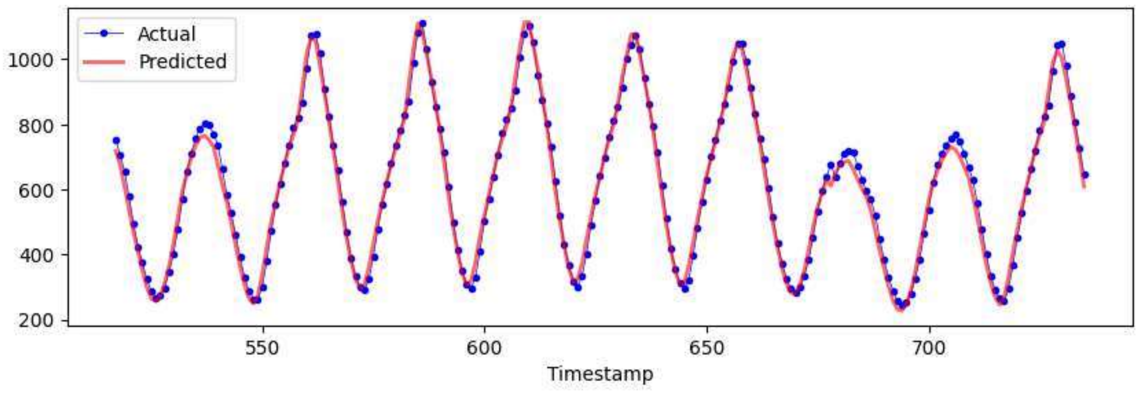
```
# Making predictions

# y_train_pred = model.predict(x_train).reshape(-1,1)
y_test_pred = model.predict(x_test).reshape(-1,1)

print( y_test_pred.shape)
```

```
(219, 1)
```

After tuning the model on the above defined hyperparameters, with the minimum error, we get the following prediction:



Mean Squared Error: 694.67
Mean Absolute Percentage Error: 4.19373217 %