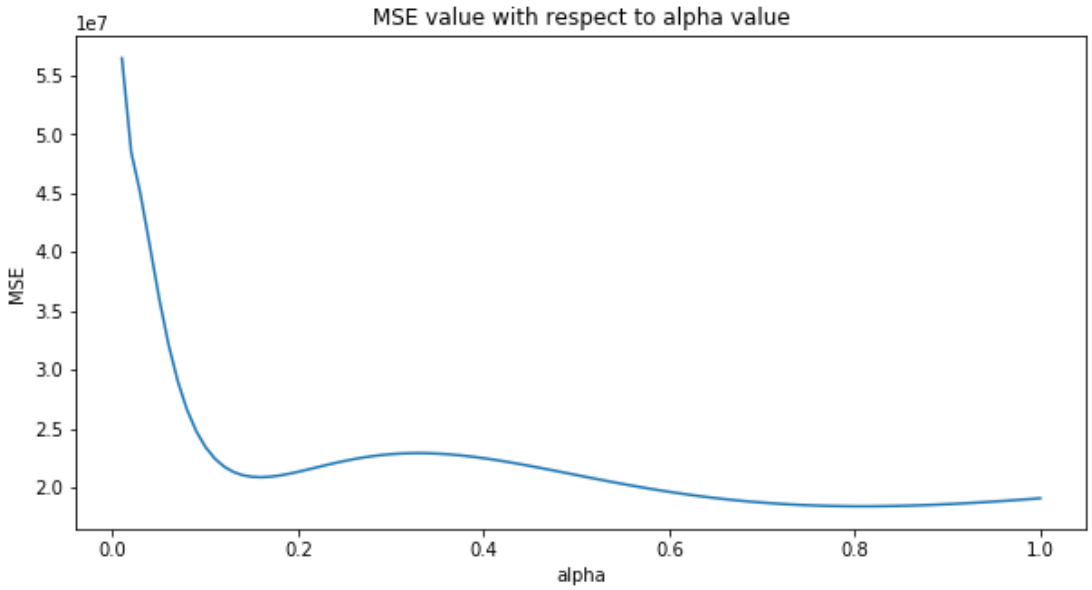
The prediction made by exponential smoothing model is straight line based on the last value we simulated Therefore, the MSE vs alpha plot varies a little when alpha value is larger than 0.7.



The minimum error is $1.839*10^7$. The corresponding alpha value is 0.81.

```
MSE       1.839193e+07
alpha     8.101010e-01
Name: 89, dtype: float64
```

Code:

```
import numpy as np                                    # vectors and matric
es
import pandas as pd                                   # tables and data ma
nipulations
import warnings                                       # There would be no
warnings
warnings.filterwarnings('ignore')
import hydrofunctions as hf
observation = hf.NWIS('03335500', 'iv', start_date='2019-01-01',end_d
ate='2019-06-30')
observation.get_data()
Timeseries = observation.df()
Timeseries.columns=["discharge","flag"]
Timeseries.head()
Timeseries.to_csv("Timeseries.csv",sep = ',')
Daily = Timeseries.resample('D').mean()
```

In [ ]:

```
import matplotlib.pyplot as plt
%matplotlib inline
Time = pd.to_datetime(Daily.index)
fig, ax = plt.subplots(figsize = (15,7))
plt.plot(Time, Daily.discharge)
ax.set(xlabel='Date',
       ylabel='Discharge Value (cfs)',
       title='Wabash River at Lafayette Station 2019');
plt.show()
```

In [ ]:

```
class Exp_Smoothing:
    """
    Exponential Smoothing model
    # series - initial time series
    # alpha  - exponential smoothing parameter
    # n_preds - prediction horizon
    """
    def __init__(self, series, alpha, n_preds):
        self.series = series
        self.alpha  = alpha
        self.n_preds= n_preds
    def exponential_smoothing(self):
        self.result = []
        for i in range(len(self.series)+self.n_preds):
            if i == 0: # components initialization
                smooth = self.series[0]
                self.result.append(self.series[0])
                continue
            if i >= len(self.series): # predicting
                val_pre = self.result[i-1]
                smooth  = self.alpha*val_pre + (1-self.alpha)*val_pre
                self.result.append(smooth)
            else:
                val_obs = self.series[i-1]
                val_pre = self.result[i-1]
                smooth = self.alpha*val_obs + (1-self.alpha)*val_pre
                self.result.append(smooth)
```

In [ ]:

```
from sklearn.metrics import mean_squared_error
def Train_Score(params, series, validsize, loss_function=mean_squared
_error):
    """
        Returns error
        param   - parameter for optimization
        series   - timeseries dataset
        validsize- size of validation dataset
    """
    values = series.values
    alpha  = params
    # split the daily dataset into training set and validation set
    train  = values[:-validsize]
    valid = values[-validsize:]

    # do the exponential smoothing and prediction by prediction model

    model  = Exp_Smoothing(series=train, alpha=alpha, n_preds=validsi
ze)
    model.exponential_smoothing()
    # select prediction results by prediction model
    predict= model.result[-validsize:]

    # find MSE by mean_squared_error function
    error = loss_function(predict,valid)
    return error
```

In [ ]:

```
# prediction horizon is 5
predict_days = 5

# make an empty error list to store MSE
Error = []
# generate alpha value starts from 0.01 to 1.0 (100 values)
alpha = np.linspace(0.01, 1.0, num=100)

# loop for calculating errors corresponding to different alpha
for i in alpha:
```

```
    error = Train_Score(i, Daily.discharge, predict_days, loss_functi
on=mean_squared_error)
    # save the error
    Error.append(error)
```

In [ ]:

```
"""
Find minimum error and correspondin alpha
"""

# put error and alpha into a dataframe
Err_alpha = pd.DataFrame(list(zip(Error,alpha.tolist())),columns=['MS
E','alpha'])
# sort the dataframe to get minimum error and corresponding alpha
Err_alpha_sorted = Err_alpha.sort_values(by=['MSE'])
Err_alpha_sorted.iloc[0]
```

In [ ]:

```
"""
Plot the MSE vs alpha plot
"""
# Plot the result
fig, ax = plt.subplots(figsize = (10,5))
plt.plot(alpha, Error)
ax.set(xlabel='alpha',
       ylabel='MSE',
       title='MSE value with respect to alpha value');
plt.show()
```