Nome: Nathana Facion

Exercício 7
Aprendizado de Máquina

---

1. Bibliotecas:

```
import numpy as np
import pandas as pd
import sys
import matplotlib.pyplot as plt
from sklearn import svm
from matplotlib import gridspec
```

2. Funções usadas:

```
# Cria o grafico com media e desvio padrao.
def graphMeanStd(fileName, imageName):
  dateparse = lambda dates: pd.datetime.strptime(dates, "%Y-%m-%d %H:%M:%S")
  timeseries = pd.read_csv(fileName, index_col='timestamp', date_parser=dateparse)
  rolmean = pd.rolling_mean(timeseries, window=50)
  rolstd = pd.rolling_std(timeseries, window=50)
  orig = plt.plot(timeseries, color='blue', label='Original')
  mean = plt.plot(rolmean, color='red', label='Media')
  std = plt.plot(rolstd, color='black', label='Desvio Padrao')
  plt.legend(loc='best')
  plt.title(imageName)
  plt.savefig(imageName + ".png")
  plt.tight_layout()
  #plt.show()

# soma os valores das ondas
def wavesSum(values):
  sum = 0
  for v in values:
    if  not np.isnan(v):
      sum += v
  return sum

# Detecta janela de anomalia.
def wavesGraph(imageName ,waves, data, type, step):
  plt.figure()
  n_graph_rows = 3
  n_graph_cols = 3
```

```
  graph_n = 1
  wave_n = 0
  greater_sum = [0 if type == 'Maior' else sys.maxint]
  greater_wave = 0
  for _ in range(n_graph_rows):
    for _ in range(n_graph_cols):
      axes = plt.subplot(n_graph_rows, n_graph_cols, graph_n)
      axes.set_ylim([-100, 150])
      mean = pd.rolling_mean(waves[wave_n], window=100)
      std = pd.rolling_std(waves[wave_n], window=100)
      plt.plot(mean, label='Media')
      plt.plot(std, label='Desvio Padrao')
      sum = wavesSum(waves[wave_n])
      if type == 'Maior':
        if sum > greater_sum:
          greater_sum = sum
          greater_wave = wave_n
      else:
        if sum < greater_sum:
          greater_sum = sum
          greater_wave = wave_n
      plt.plot(waves[wave_n], label='Original')
      graph_n += 1
      wave_n += step

  plt.tight_layout()
  #plt.show()

  # Plota as diferentes janelas criadas para analisar
  plt.savefig(imageName + "-Fases.png")
  plt.clf()

  # debug
  # plt.plot(waves[greater_wave])
  # plt.savefig("Teste" + ".png")
  #plt.show()

  oneClass(imageName,waves[greater_wave],data)


# Essa funcao faz com que o arquivo seja dividido em segmentos
# esses segmentos serão usados para detectarmos a janela de anomalia
def waveWindows(fileSerie1, imageName, window, inters, type):
  dateparse = lambda dates: pd.datetime.strptime(dates, "%Y-%m-%d %H:%M:%S")
  data = pd.read_csv(fileSerie1, index_col='timestamp', date_parser=dateparse)
  dataReturn  = data

  # Duas questoes a serem tratadas sao:
  #  qual o valor de N
  segment_len = window
  # e quanto de interssecao entre os trechos.
  slide_len = inters

  segments = []
  for start_pos in range(0, len(data), slide_len):
    end_pos = start_pos + segment_len
    segment = np.copy(data[start_pos:end_pos])
    if len(segment) != segment_len:
      continue
    segments.append(segment)
```

```
  windowed_segments = []
  window_rads = np.linspace(0, np.pi, segment_len)
  window = np.sin(window_rads) ** 2

  for segment in segments:
    windowed_segment = np.copy(segment) * window
    windowed_segments.append(windowed_segment)

  wavesGraph(imageName, segments, dataReturn, type, step=3)
```

3. Algoritmo detecta anomalia em trechos

```
    # Algoritmo usado para detectar anomalias no trecho : SVM One Class - visto em sala de aula
def oneClass (imageName, X_outliers, X_train):

  clf = svm.OneClassSVM(nu=0.1, kernel="rbf", gamma=0.1)
  clf.fit(X_train)
  y_pred_train = clf.predict(X_train)
  y_pred_outliers = clf.predict(X_outliers)
  n_error_train = y_pred_train[y_pred_train == -1].size
  n_error_outliers = y_pred_outliers[y_pred_outliers == 1].size

  # Divide o grafico em duas telas
  fig = plt.figure(figsize=(16, 6))
  gs = gridspec.GridSpec(1, 2, width_ratios=[4, 1])

  ax1 = plt.subplot(gs[0])
  ax1.plot(X_train)
  ax1.set_title('Serie')

  ax2 = plt.subplot(gs[1])
  ax2.plot(X_outliers,color='green', markersize=5)
  ax2.set_title('Anomalia')

  fig.text(0.5, 0.04, "Numero de erros no treino: %d/4033 ; Numero de erros na janela de  anomalia: %d/1200" %
(n_error_train, n_error_outliers), ha='center', va='center')

  plt.savefig(imageName + ".png")
  plt.axis('tight')
  plt.clf()
  #plt.show()
```

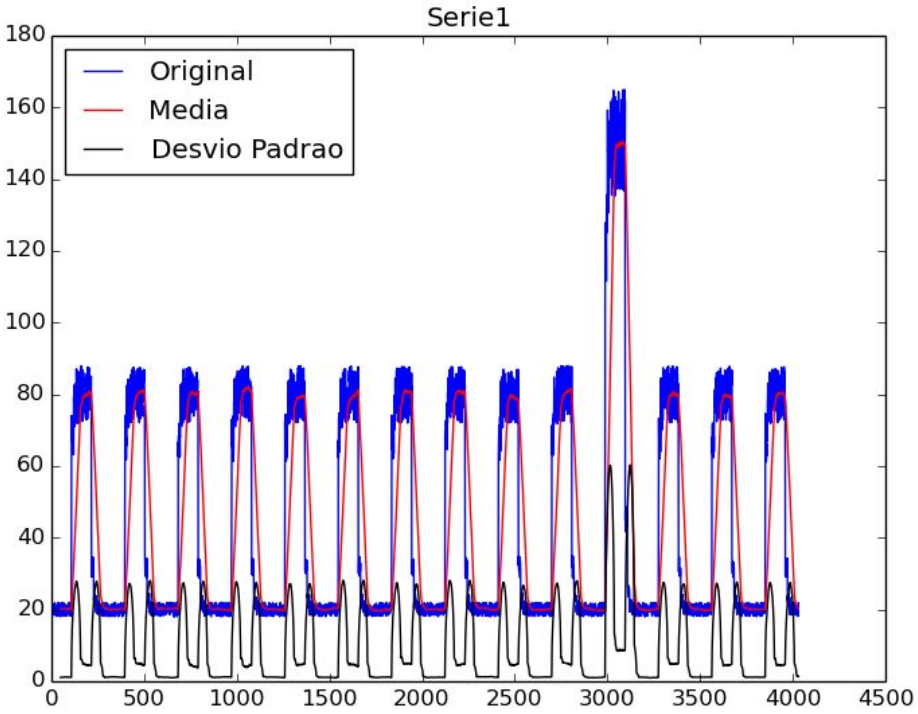
4. Main

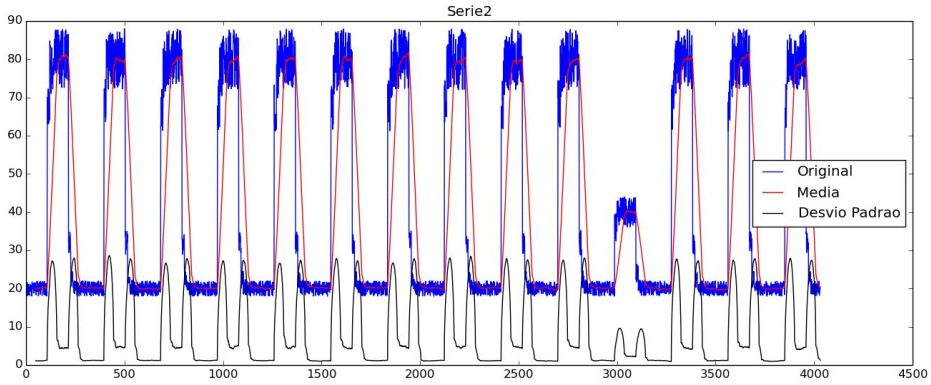
```
def main():
  # leitura da serie 1 ------------------------------------
  fileSerie1 = "//home//nathana//AM//exercicio7//serie1.csv"

  # Plota medias e desvio padrao
  graphMeanStd(fileSerie1, 'Serie1')

  # Encontra janela com anomalia
  waveWindows(fileSerie1, 'Serie1- window',1200,100,'Maior')

  # leitura da serie 2 ------------------------------------
  fileSerie2 = "//home//nathana//AM//exercicio7//serie2.csv"

  # Plota medias e desvio padrao
  graphMeanStd(fileSerie2, 'Serie2')

  # Encontra janela com anomalia
  waveWindows(fileSerie2,'Serie2- window',1200,100, 'Menor')

  # leitura da serie 3 ------------------------------------
  fileSerie3 = "//home//nathana//AM//exercicio7//serie3.csv"

  # Plota medias e desvio padrao
  graphMeanStd(fileSerie3, 'Serie3')
  waveWindows(fileSerie3,'Serie3- window',1200,100, 'Menor')

  # leitura da serie 4 ------------------------------------
  fileSerie4 = "//home//nathana//AM//exercicio7//serie4.csv"

  # Plota medias e desvio padrao
  graphMeanStd(fileSerie4, 'Serie4')

  # Encontra janela com anomalia
  waveWindows(fileSerie4,'Serie4- window',1200,100,'Maior')

  # leitura da serie 5 ------------------------------------
  fileSerie5 = "//home//nathana//AM//exercicio7//serie5.csv"

  # Plota medias e desvio padrao
  graphMeanStd(fileSerie5, 'Serie5')

  # Encontra janela com anomalia
  waveWindows(fileSerie5,'Serie5- window',1200,100,'Maior')

if __name__ == '__main__':
  main()
```

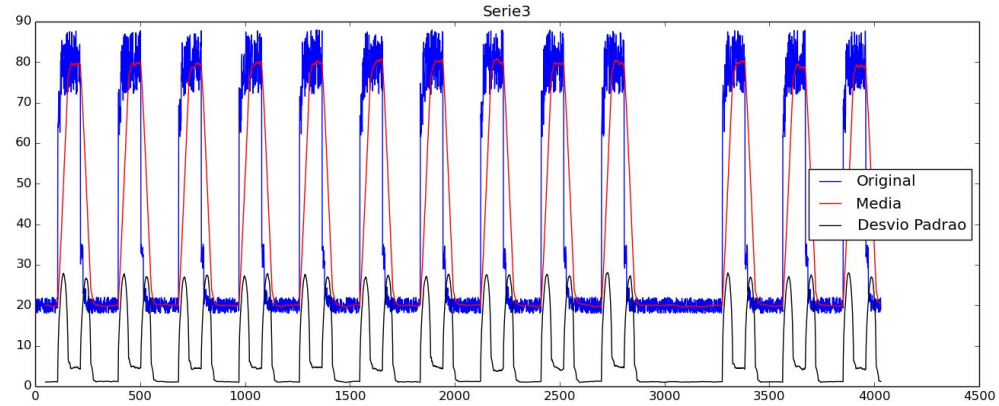
5. Respostas

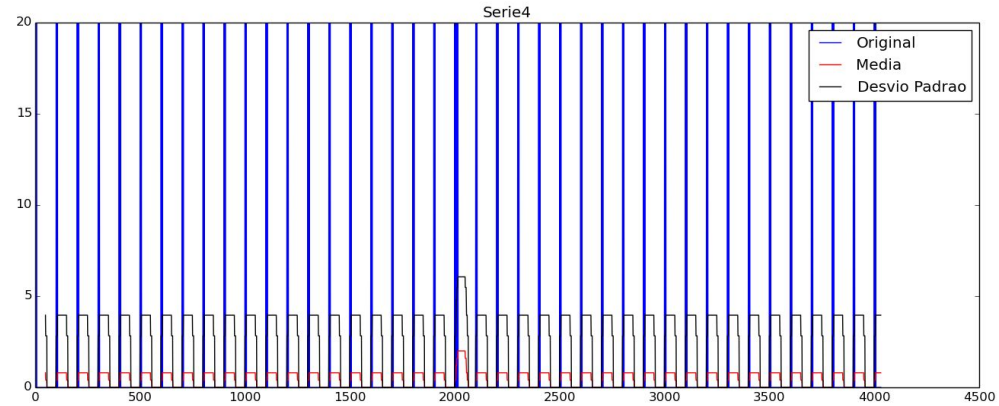
1) A ideia central é definir um trecho da série como N pontos seguidos, e representar cada trecho. Eu acho que apenas a média e desvio padrão dos valores no trecho, ou media, maximo e minimo sera suficiente. Assim, cada trecho sera representado por 2 (media e desvio padrão) ou 3 (média, máximo, mínimo) dimensões.

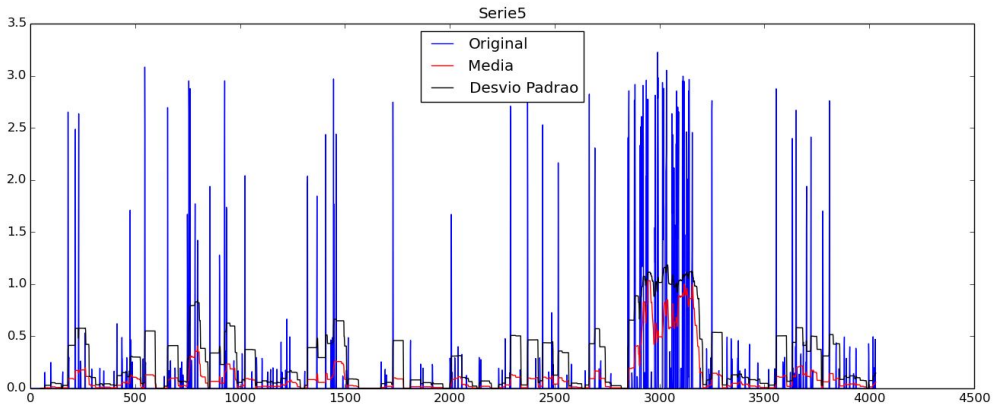
Resposta: Segue cada gráfico com média e desvio padrão. Serão usados posteriormente para análise de anomalias em janelas. A média será adotada para identificar as anomalias, em algumas séries a maior média e em outras a menor média.




Serie2
Original
Media
Desvio Padrao


Serie3
Original
Media
Desvio Padrao


Serie4
Original
Media
Desvio Padrao



2) Duas questões a serem tratadas são qual o valor de N e quanto de interseção entre os trechos.

Respostas: N = 1200 e Interseção= 100 . Segue os gráficos com as Janelas sendo usadas para analisar as anomalias em trechos. As séries 4 e 5 tem pontos muito próximo portanto nos gráficos abaixo ficam ruim de visualizar .Entretanto na questão 1 fica mais claro onde estão as anomalias.

Gráfico: Série 1 Vermelho: Original Azul:Média Verde: Desvio Padrão.

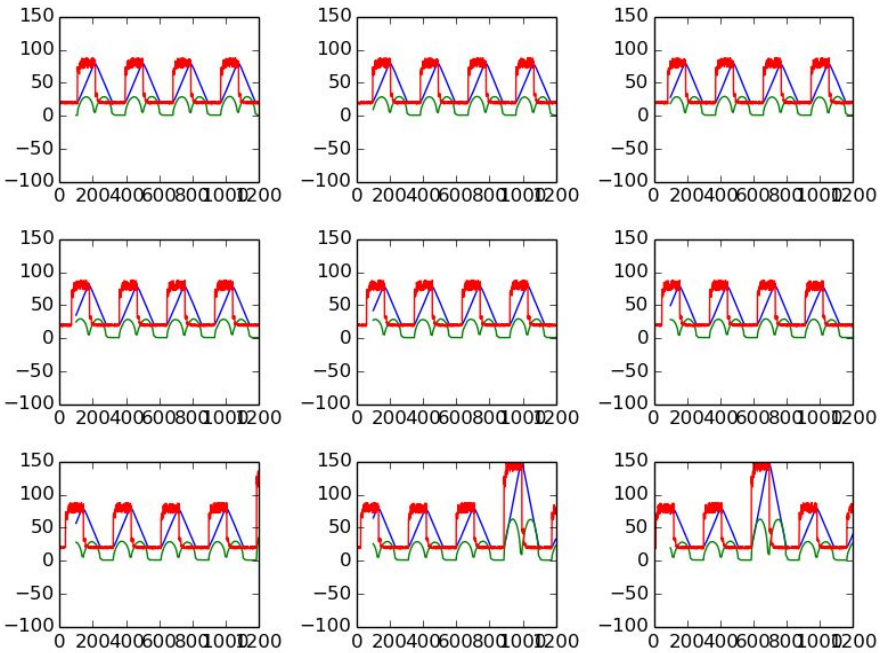


Gráfico: Série 2 Vermelho: Original Azul:Média Verde: Desvio Padrão.

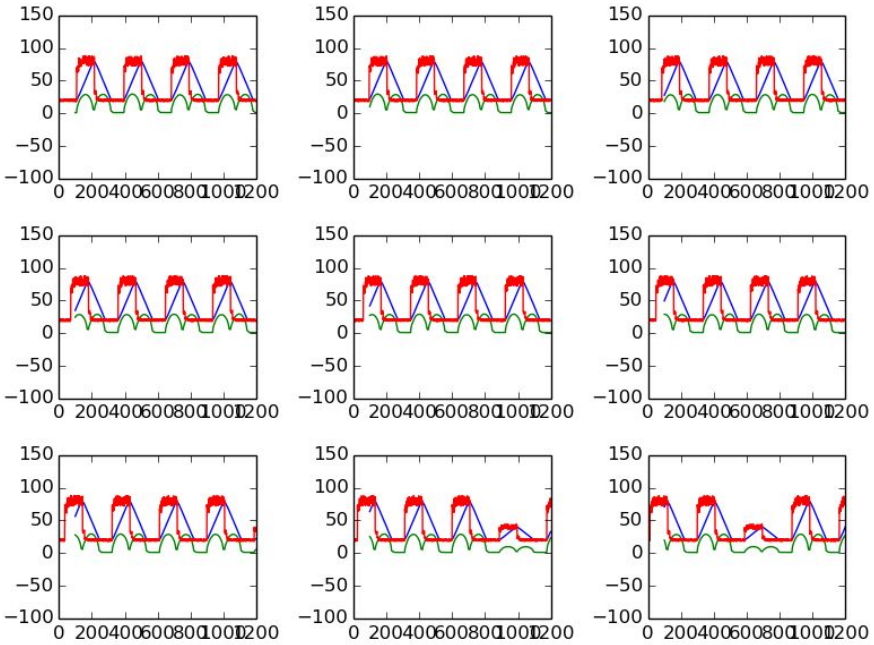


Gráfico: Série 3 Vermelho: Original Azul:Média Verde: Desvio Padrão.

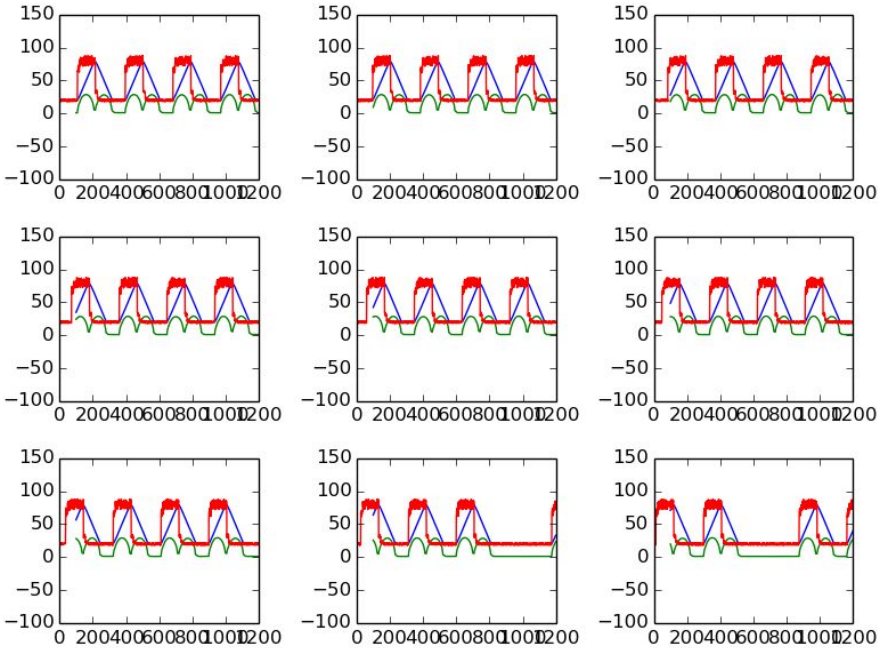


Gráfico: Série 4 Vermelho: Original Azul:Média Verde: Desvio Padrão.

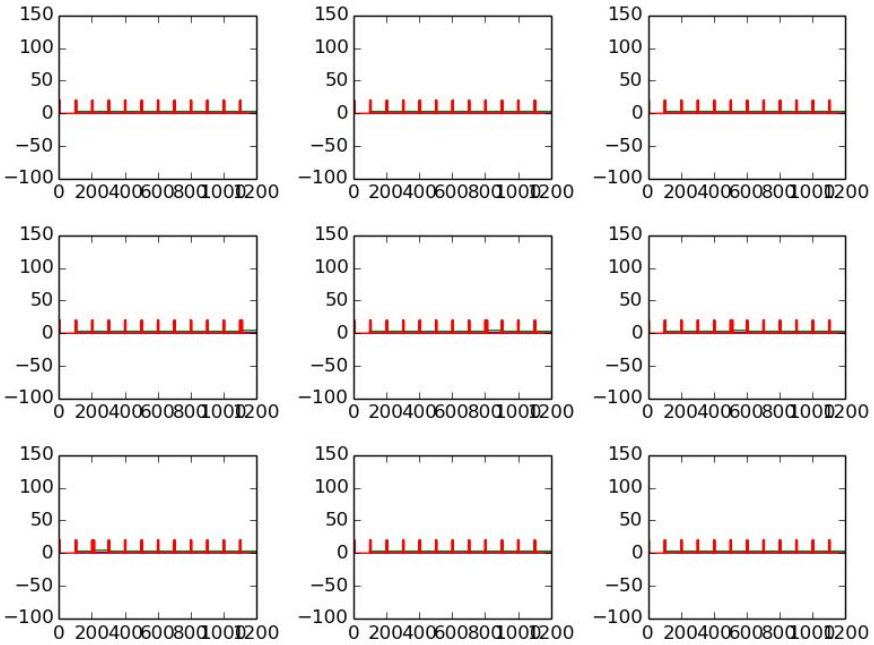
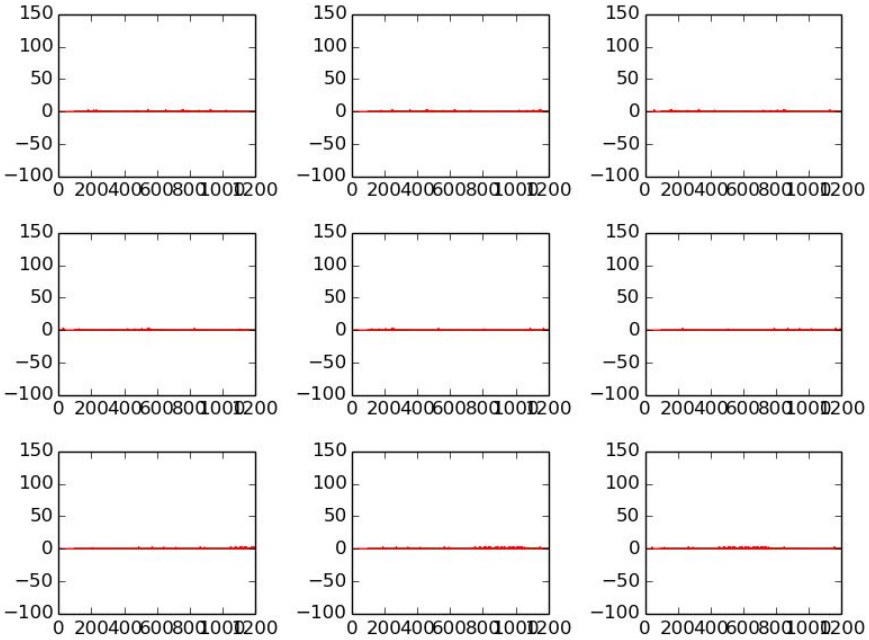
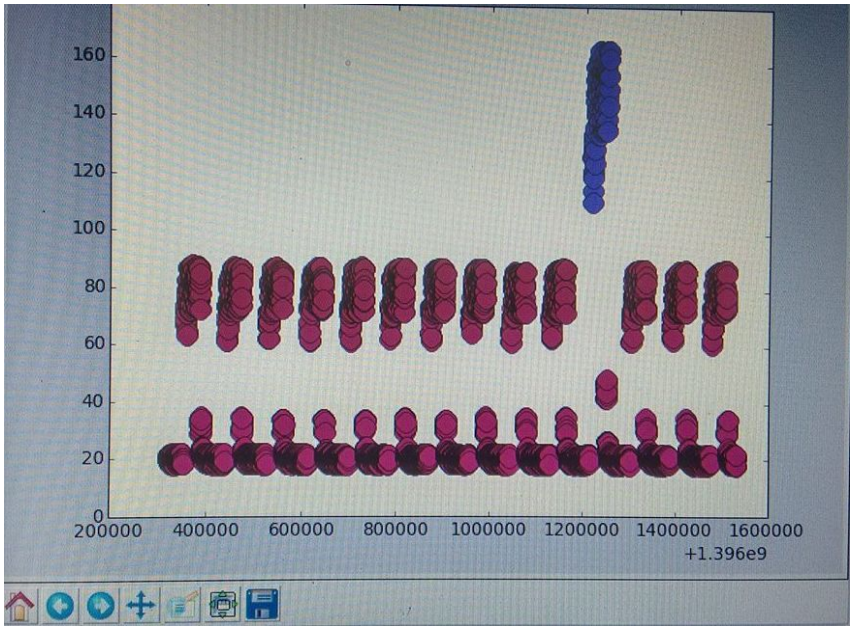


Gráfico: Série 5 Vermelho: Original Azul:Média Verde: Desvio Padrão.

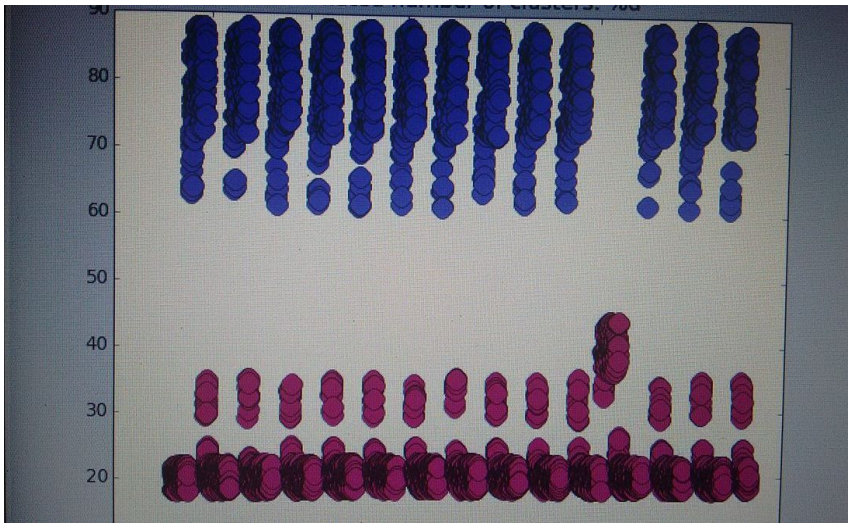


3) Com a representação de cada trecho, o problema é encontrar o/os trechos anômalos. E aqui as várias alternativas discutidas em classe sao aplicáveis: distancia ao k-esimo vizinho, dados em subespaços diferentes, dados em região de baixa densidade, etc.

Resposta: Algoritmo utilizado é SVM One Classe, que se encontra na aula de anomalia. Além deste foi testando o DBSCAN. Entretanto o resultado foi bom apenas para o primeiro caso:



O segundo caso já teve um resultado ruim:



Segue o código utilizado para esse outro método:

```
def graphDBSCAN(X,labels,core_samples_mask):
  # Black removed and is used for noise instead.
```

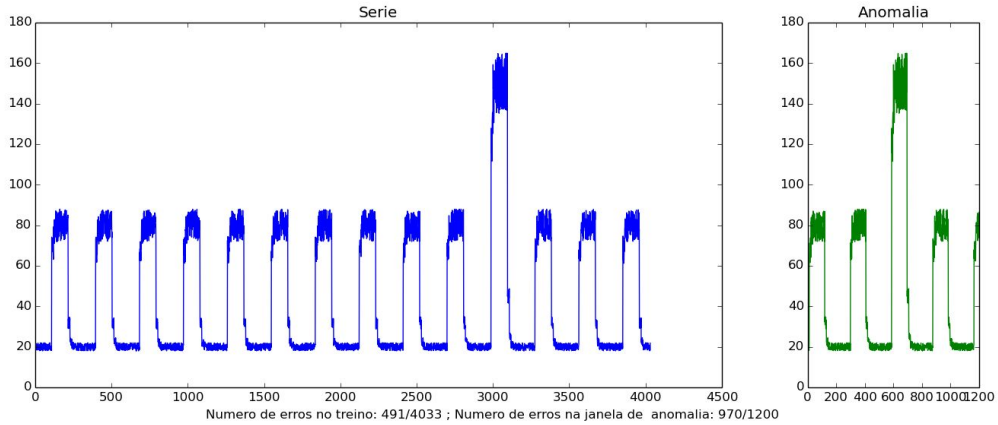
```
    unique_labels = set(labels)
    X = np.array(X)
    colors = plt.cm.Spectral(np.linspace(0, 1, len(unique_labels)))
    for k, col in zip(unique_labels, colors):
        if k == -1:
            # Black used for noise.
            col = 'k'
        class_member_mask = (labels == k)
        print 'cor',col,'k',k
        xy = X[class_member_mask & core_samples_mask]
        #print 'xy',xy
        plt.plot(xy[:,0], xy[:,1], 'o', markerfacecolor=col,markeredgecolor='k', markersize=14)

    plt.title('Estimated number of clusters:')
    plt.show()

def metrics(X,labels):
    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
    print('Estimated number of clusters: %d' % n_clusters_)

def AlgorDBSCAN(fileName,eps,min_samples ):
    # Pre - processamento
    X = process(fileName)
    stscaler = StandardScaler().fit(X)
    data = stscaler.transform(X)

    # Aplicando DBSCan
    dbsc = DBSCAN(eps = eps,min_samples = min_samples, metric = 'chebyshev' ).fit(data)
    labels = dbsc.labels_

    # Descobrindo estimativa de cluster
    metrics(X,labels)

    # Criando grafico
    print 'labels:',labels
    core_samples = np.zeros_like(labels, dtype=bool)
    core_samples[dbsc.core_sample_indices_] = True
    graphDBSCAN(X,labels,core_samples)


# Pre processa dados
def process(fileName):
    data = open(fileName)
    dataCsv = pd.read_csv(data, sep=',', header=None)[1:]
    deleteLine = []
    for i in range(dataCsv.shape[0]):
        col = dataCsv[0][i+1]
        try:
            dataCsv[0][i + 1] = time.mktime(datetime.datetime.strptime(col, "%Y-%m-%d %H:%M:%S").timetuple())
        except ValueError as e:
            print('Erro de data: dado ignorado linha', i+1)
            deleteLine.append(i)
            continue

    dataCsv = dataCsv.drop(dataCsv.index[deleteLine])
    return dataCsv
```

4) Discuta a sua solução para o problema, e mostre o resultado do seu algoritmo nas series temporais de 1 a 4. Pense numa forma de mostrar os trechos anômalos no plot das séries temporais.

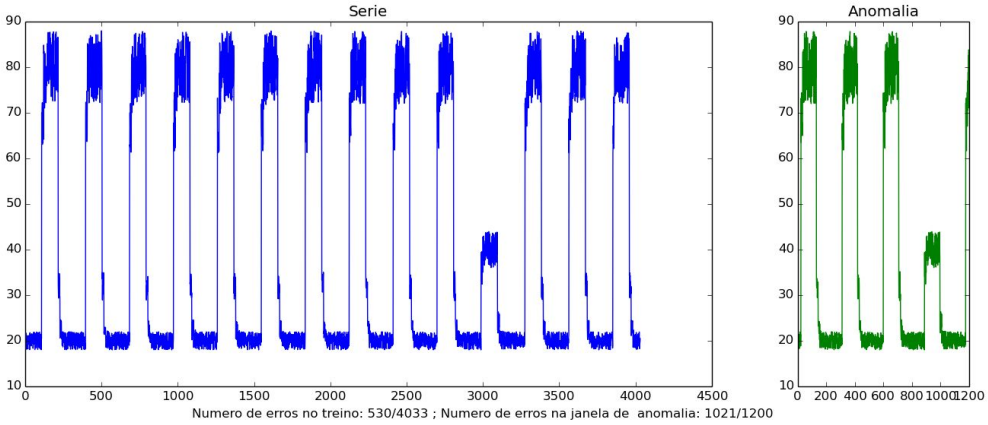
Resposta: Minha solução foi dividir os dados em segmentos de 1200 com interseção de 100. Após separar em segmentos, esses segmentos são analisados por meio da média (em algumas séries foi a maior média e em outras a menor média). Então o segmento com a maior/menor média é passado para o algoritmo SVM One Class, junto aos dados originais. Por meio da realização de fit nos dados de treinamento e do segmento com as anomalias conseguimos identificar o quanto o trecho é anômalo.

Em seguida os dados são plotados, como no trecho abaixo com os dados originais e o segmento que contém a anomalia. Abaixo dos gráficos temos uma legenda que informa o quanto os dados originais são anômalos e o quanto o segmento com anomalias é anomalo.

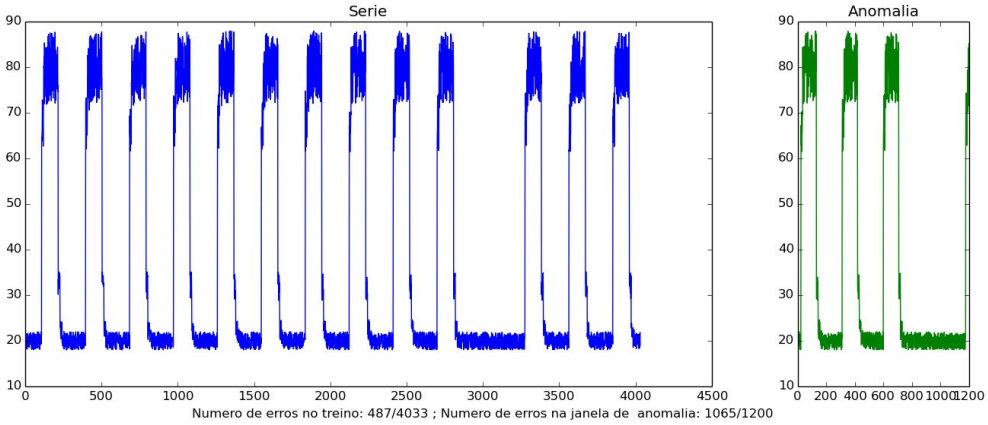
Série 1:



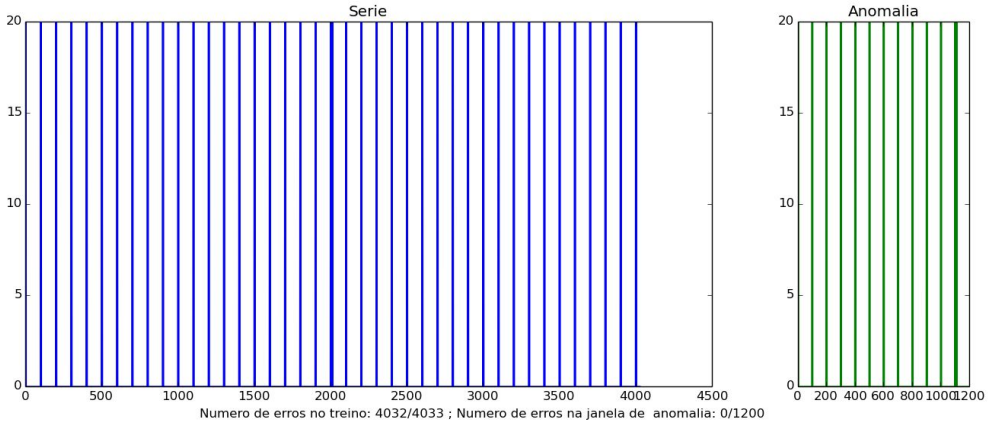
Série 2:



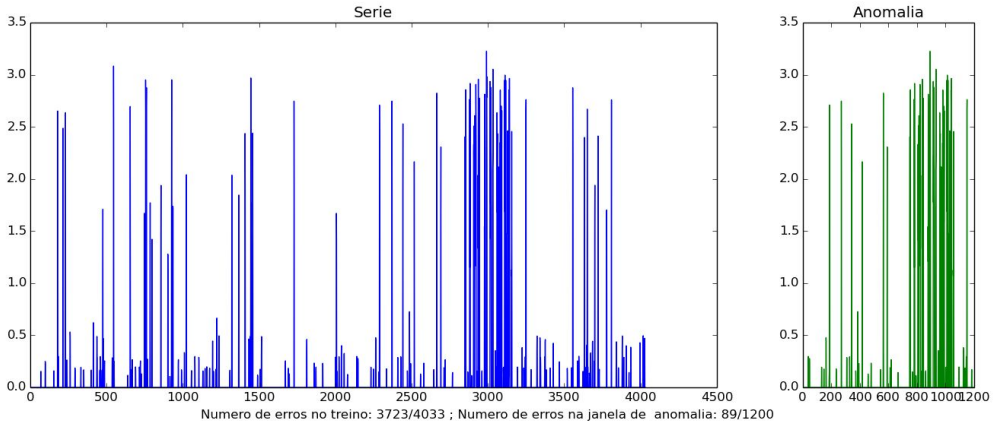
Série 3:



Série 4:



Série 5:



5) Finalmente rode o seu algoritmo na série temporal 5 (onde nao é nada claro o que é anômalo e nao anômalo) e mostre e discuta os resultados.

Resultado: A maior média conseguiu identificar uma concentração maior de dados em um trecho, com isso definiu esse trecho como anômalo. Entretanto o próprio SVM One Class teve dificuldade para identificar a quantidade de erros no treino e na janela de anomalia. De acordo com o algoritmo o trecho acima não seria o que tem anomalia pois apresenta apenas 88 erros.