DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Relatório do Projeto de LAPR1

Análise de Séries Temporais.

Equipa *07/ Os Ases 1DKL*
1190633_Gonçalo Jordão (1DL)
1190699_João Osório (1DK)
1190778_Miguel Silva (1DK)
1190995_Ricardo Mesquita (1DL)
1191045_Rui Soares (1DK)

Docente(s)/Orientador(es)
Carlos Ferreira (CGF)
Ana Barata (ABT)
Sandra Luna (SLU)

Cliente
Carlos Ferreira (CGF)

Unidade Curricular
Laboratório/Projeto I

Dezembro 2019

isep Instituto Superior de Engenharia do Porto

Índice

1. Introdução

O presente relatório foi realizado no âmbito da disciplina de Laboratório/Projeto I, pertencente ao curso de Engenharia Informática, do Instituto Superior de Engenharia do Porto.

Neste documento será apresentada a informação relativa ao desenvolvimento do projeto da disciplina de Laboratório/Projeto I, sobre a análise de séries temporais. Este trabalho foi realizado em linguagem JAVA e vai servir como base à realização do projeto. Neste relatório serão mencionados os objetivos do projeto, metodologias usadas, dificuldades encontradas, resultados obtidos e a análise teórica das séries temporais.

Esta metodologia passa por um planeamento, com identificação dos objetivos propostos e a construção de um código em Java, que por sua vez foi devidamente elaborado em grupo.

Esse mesmo código, juntamente com todos os dados fornecidos, permitiu-nos obter todos os resultados para a conclusão deste projeto

Este relatório vai dar uma maior ênfase à metodologia de trabalho usada, ao estudo feito nas séries temporais e os seus consecutivos benefícios no mercado de trabalho e ao desenvolvimento e implementação do projeto, onde se falará também dos testes realizados para verificar os resultados dados ao longo da execução do programa. Terá também uma parte reservada aos resultados e, por último, uma conclusão da totalidade do projeto.

2. Metodologia de Trabalho

Neste tópico vão ser abordados vários pontos relacionados com a metodologia de trabalho do nosso grupo nomeadamente o papel do EduScrum no nosso trabalho, assim como várias informações do mesmo. Mencionamos também como foi feito o planeamento e distribuição das nossas tarefas bem como uma reflexão crítica da dinâmica do nosso grupo.

2.1 EduScrum no desenvolvimento do Projeto:

O EduScrum consiste numa estrutura que treina estudantes, onde os professores atribuem a responsabilidade do processo de aprendizagem aos alunos. É também um modelo de ensino que permite aos alunos resolver problemas complexos de maneiras produtivas e criativas, alcançando assim metas de aprendizagem e de crescimento pessoal. O EduScrum é um método que valoriza mais aquilo que tem de ser feito e não a maneira de como é feito (sendo assim possível aplicar diversas técnicas diferentes para se chegar ao mesmo fim), desafiando assim os estudantes a auto-organizarem-se e a realizarem um trabalho com qualidade dentro de um período limitado, com os objetivos de aprendizagem bem definidos.

A metodologia scrum apresenta um planeamento de todas as tarefas, e a designação dessas mesmas para com todos os elementos que constituem o grupo. Reuniões e discussões ao longo de todo o desenvolvimento do projeto, são aspetos essenciais para que o grupo se mantenha em sintonia e organizado na sua realização, e com isto, a criação de um "backlog" (listagem e atribuição das tarefas) a cada elemento.

Quanto a estrutura do EduScrum, esta é equivalente à do Scrum original, todavia voltada para a educação, ou seja, consiste em equipas e cada membro da equipa com o seu papel, evento, e regras associados. Cada membro tem o seu propósito específico e é essencial para o sucesso da equipa e também para o sucesso do uso do método EduScrum.

O EduScrum, tal como o Scrum, baseia-se na teoria do controle de processos empíricos. O empirismo afirma que o conhecimento vem de experiência e de se tomar decisões com base no que se é conhecido.

Para além disso, o EduScrum baseia-se em três pilares importantes para o controle de processos empíricos: transparência, inspeção e adaptação.

2.2 Planeamento e distribuição de tarefas:

Refletir sobre o plano do trabalho na sua totalidade, a organização em equipa, a distribuição de tarefas; referir as ferramentas utilizadas (de planeamento e de comunicação) e sua utilidade.

Distribuição de tarefas:

Durante o início das aulas, reuníamo-nos e definíamos o que cada um teria de realizar nessa aula e também os objetivos que tínhamos de alcançar durante o período letivo, e caso isso não aconteça, resolvíamos os problemas em casa, e através de uma das ferramentas de comunicação tirávamos duvidas uns com os outros e enviamos os nossos progressos.

Ao longo destas últimas semanas de trabalho foram utilizadas várias ferramentas de trabalho tais como:

- **Trello:** Uma aplicação em web que nos permite gerenciar todo o nosso projeto bem como delinear o caminho a seguir até a obtenção do projeto final. No nosso caso foi também uma aplicação que nos ajudou na atribuição de tarefas diariamente a todos os elementos do grupo e também que nos ajudou a perceber o ponto de situação do nosso trabalho.

- **Bitbucket:** Um serviço especializado na agregação de projetos, utilizando o Git (Sistema de controlo para registo de histórico). Com a utilização deste site foi possível, partilhar o devido repositório, criado pelo grupo, para mais fácil acesso de todos os membros ao trabalho e aos ficheiros desenvolvidos por cada um de nós.

- **Sourcetree:** Um cliente em Git desenvolvido para a organização de projetos. Esta aplicação permite efetuar upload de ficheiros para o Bitbucket, facilitando assim a transferência do conteúdo do trabalho entre os vários elementos do grupo.

- **Netbeans:** Ambiente de desenvolvimento utilizado para programar em linguagem Java e no qual foi desenvolvido todo o trabalho.

- **Javaplot e Gnuplot**: Ferramentas utilizadas para o desenvolvimento dos gráficos com a ajuda do programa Netbeans;

Para toda esta gestão do trabalho foram utilizadas também algumas ferramentas de comunicação tais como:

- **Reuniões Presenciais:** Reuniões estas que permitiam o diálogo entre todos os elementos do grupo, para uma melhor compreensão de todos os pontos de vista e a tentativas de chegar a um consenso final que fosse o melhor para o projeto em si.

- **Diversas aplicações (WhatsApp, Outlook, Discord, etc…):** Aplicações estas que permitiam a comunicação a distância entre todos os elementos do grupo, para tirar qualquer dúvida em relação ao trabalho ou até mesmo para discutir alguns pontos de vista diferenciados em relação a determinados assuntos.

2.3 Reflexão crítica sobre a dinâmica do grupo:

Acreditamos que o grupo trabalhou bem, tentamos sempre dar o nosso melhor no trabalho, quer nas tarefas individuais quer nas coletivas. Dividimos algumas tarefas pelos membros do grupo, sendo que cada um deu o seu melhor para elaborar a parte do trabalho que lhe estava destinada, e obviamente, que as partes que suscitam mais dúvidas foram abordadas por todos os membros, para ser mais fácil e eficaz a sua resolução. Assim achamos que todos os membros do grupo se esforçaram ao máximo para fazer o trabalho, embora não tivéssemos tido o tempo desejado para melhorarmos algumas partes do mesmo.

Apesar de tudo e todos os contratempos foi uma dinâmica de grupo muito boa e para levar como exemplo.

3. Análise de Séries Temporais

3.1 Sobre a Análise de Séries Temporais:

Uma série temporal é uma sequência de observações ordenada cronologicamente que, em geral, são recolhidos em intervalos regulares. A análise de séries temporais pode ser aplicada a qualquer variável que muda ao longo do tempo e, de um modo geral, as observações mais próximas tem valores mais próximos que aqueles valores mais distantes.

A análise de séries temporais é de grande utilidade em vários domínios como por exemplo as Finanças, Meteorologia, Energia, entre outros, sendo que do seu processamento podem resultar ganhos significativos para o conhecimento do negócio ou planeamento de atividades

Para extrair conhecimento das séries temporais existe um conjunto alargado de métodos, técnicas e ferramentas que podem ser utilizados. Alguns destes métodos exigem conhecimentos avançados que são apenas adquiridos através de formação avançada ao nível de formação superior especializada. Outros métodos, como os que serão explorados neste projeto, são mais simples e podem ser estudados e implementados por qualquer aluno que frequenta um curso superior de engenharia ou ciências. Entre estes estão métodos e técnicas básicos que permitem analisar uma série temporal em diferentes resoluções, filtrar/suavizar uma série e prever valores futuros utilizando modelos.

3.2 Análise utilizando diferentes resoluções:

As observações que constituem uma série temporal podem ser recolhidas com elevada frequência, para permitir monitorizar e analisar com rigor ocorrências, ou com menor frequência, quando o estado de um sistema ou processo praticamente não é alterado ao longo do tempo e uma amostra recolhida com grande espaçamento é suficiente para analisar a evolução de um sistema ou processo.

Em geral, e atendendo à tecnologia disponível, recolhem-se dados com elevada frequência que depois são transformados para a resolução necessária ao objetivo e problema que se pretende resolver. Por exemplo, se pretendemos analisar o consumo de eletricidade mensal de uma casa e estamos a recolher dados que representam o consumo a cada hora, então teremos de somar o consumo de todos os dias e horas de um determinado mês para calcular o consumo total nesse mês.

Uma técnica relevante para analisar uma série temporal é a filtragem. Ao filtrar os dados podemos remover ruído e identificar tendências. Entre as técnicas de filtragem mais utilizadas e simples estão a Média Móvel Simples e a Média Móvel Exponencialmente Pesada (que são as técnicas utilizadas neste projeto).

A **Média Móvel Simples** é definida pela equação:

$$y_i = \frac{1}{n} \sum_{k=0}^{n-1} x_{i-k},$$

Onde xi são os termos que representam a série original, yi é a série resultante da aplicação do filtro (da suavização) e n é a ordem da média móvel.

A **Média Móvel Exponencialmente Pesada** é definida pela equação:

$$y_i = \alpha x_i + (1 - \alpha) y_{i-1},$$

Onde xi são os termos que representam a série original yi é a série resultante da aplicação do filtro (da filtragem) e α é uma constante que toma valores no intervalo]0,1]. Nas Figuras 1 e 2 são apresentadas uma série original e uma série filtrada que resulta da aplicação do modelo Média Móvel Exponencialmente Pesada, para dois valores distintos de α (0.05 e 0.5).

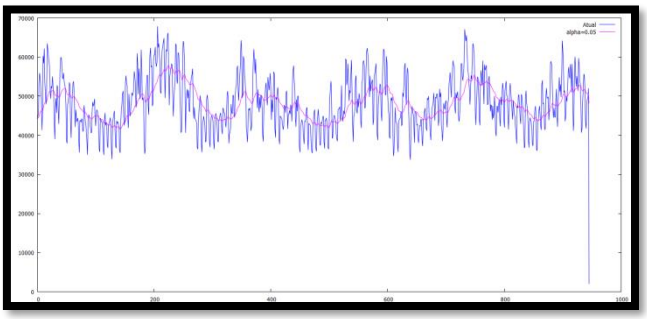

Fig. 1 – Análise de Séries – MMP α de 0.05 (Diário)

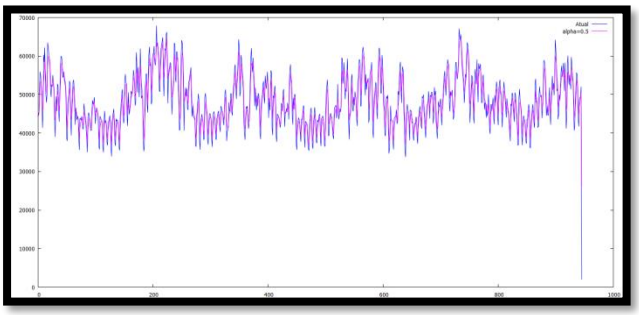

Fig. 2 – Análise de Séries – MMP α de 0.5 (Diário)

3.3- Previsão:

Para analisar uma série temporal também é possível encontrar um modelo matemático que capture o processo que gerou a série temporal e permita prever valores futuros da série utilizando o histórico de dados. Entre os modelos mais utilizados e simples estão os modelos de Média Móvel Simples e os modelos de Média Móvel Exponencialmente Pesada (que são as técnicas utilizadas neste projeto).

Para realizar uma previsão utilizando a **Média Móvel Simples** recorremos à equação:

$$y_{i+1} = \frac{1}{n} \sum_{k=0}^{n-1} x_{i-k},$$

Onde xi são os termos que representam a série original, yi+1 representa uma previsão utilizando dados históricos e n é a ordem da média móvel.

No caso da **Média Móvel Exponencialmente Pesada**, para realizar uma previsão recorremos à equação:

$$y_{i+1} = \alpha x_i + (1 - \alpha) y_i,$$

Onde xi são os termos que representam a série original, yi+1 representa uma previsão utilizando dados históricos e α é uma constante que toma valores no intervalo]0; 1].

4. Desenvolvimento e Implementação da Aplicação

4.1-Estruturação da aplicação:

Linguagem utilizada:

Neste projeto, a linguagem de programação utilizada foi *java*, visto que pretendíamos dar continuidade ao percurso da unidade curricular de APROG.

4.2-Codificação e execução dos modos:

4.2.1-Interativo:

No modo interativo a aplicação é chamada pela linha de comandos, utilizando o respetivo comando (*java -jar programa.jar –nome nome.csv*), executado por dois parâmetros, tendo o segundo parâmetro de ser, obrigatoriamente, um ficheiro *.csv*.

```
int n = 0, s = args.length;  //sendo s o número de parâmetros introduzidos pelo utilizador
```

```
if (s == 2) {
    while (opcao != 0) {
        System.out.println();
        System.out.println("=============================================================MENU=============================================================");
        System.out.println();
        System.out.println("OPÇÃO 1:  Visualisar gráfico dos consumos.");
        System.out.println("OPÇÃO 2:  Visualisar média global e distribuição de observações.");
        System.out.println("OPÇÃO 3:  Ordenar valores.");
        System.out.println("OPÇÃO 4:  Efetuar filtragens.");
        System.out.println("OPÇÃO 5:  Efetuar uma previsão.");
        System.out.println("OPÇÃO 6:  Escolher outro ficheiro.");
        System.out.println("OPÇÃO 0:  Fechar o programa!");
        System.out.println();
        System.out.print("Insira a opção que pretende: ");
        opcao = tec.nextInt();
```

Fig. 3 – Menu do programa (Modo Interativo)

Assim, se o número de parâmetros corresponder ao pretendido, e todas as verificações forem feitas com sucesso, é-nos mostrado o modo interativo com os consecutivos menus e opções.

4.2.2-Não interativo:

No modo não interativo, a interação com o utilizador é nula e assim, pretende-se que todas as funcionalidades da aplicação sejam executadas por um único comando (especificando todos os parâmetros).

Para isso, tiveram de ser feitas verificações ao número de parâmetros inseridos:

```
if (s == 12) {

    if ("11".equals(args[3]) || "12".equals(args[3]) || "13".equals(args[3]) || "14".equals(args[3]) || "2".equals(args[3]) || "3".equals(args[3]) || "4".equals(args[3])) {
        resolução = args[3];
        n++;
    }

    if ("1".equals(args[5]) || "2".equals(args[5])) {
        modelo = args[5];
        n++;
    }

    if ("1".equals(args[7]) || "2".equals(args[7])) {
        tipoOrdenação = args[7];
        n++;
    }
```

(...)

Fig. 4 – Modo Não Interativo

-Outras opções:

No caso de nenhum dos modos se verificar, é apresentada uma mensagem, informando o utilizador do respetivo erro.

4.3-Sistema Modular (Modularização):

Na realização da aplicação, utilizamos um sistema modular (ou modularizado), assim como nos foi pedido, no âmbito de subdividir o código em diferentes partes, tornando-o mais flexível e legível por parte do utilizador.

Este conceito visa reduzir a complexidade do problema, dividindo-o em sub-problemas mais simples, que podem inclusivamente ser resolvidos por equipas e grupos independentes, facilitando também a deteção de erros.

Nesta aplicação, a modularização foi utilizada para diversas tarefas: desde a criação de arrays com informação útil de ficheiros e métodos de ordenação, a objetivos mais complexos como por exemplo o cálculo de médias e previsões.

4.4-Estruturação de dados:

Na estruturação dos dados, evitamos o uso das variáveis globais, assim como nos foi pedido.

```
public static void main(String[] args) throws FileNotFoundException {
    String ficheiro, resolução = "Null", modelo = "Null",
            tipoOrdenação = "Null", parModelo = "Null", momentoPrevisão, title, escolha = "";
    int n = 0, s = args.length;
    String[] dados = new String[3];

    if (s >= 2) {
        ficheiro = args[1];

        String[][] data_horas = new String[LINHAS][2];
        int[] energia = new int[LINHAS];
        int posição, opcao = 99, qtd, qtd_aux, X_aux = 0, aux = 1, ult_aux = 1;
        double nAlpha_aux = 0;
        String data_aux1 = "";

        posição = Leitura(data_horas, energia, ficheiro);
        int[] energia_aux = new int[posição];
        int[] energia_media = new int[posição];
        String[] datasHoras_aux = new String[posição];
```

Fig. 5 – Estruturação dos dados

Com base no conhecimento que possuíamos, decidimos guardar os diferentes tipos de informação incluídos no ficheiro base (DAYTON.csv), em arrays distintos, para que posteriormente se tornasse mais simples reunir toda a informação necessária e adaptá-la ao conteúdo dos diferentes problemas que iam surgindo.

Assim, após ser selecionada a resolução pretendida teríamos a possibilidade de escolher os arrays que melhor se aplicariam ao determinado tipo de exercício.

Quanto à execução da filtragem, a informação da série original é armazenada num array, que após ser enviada para os módulos de execução de cálculos da Média Simples ou Média Exponencial Pesada (de acordo com a seleção do utilizador), cria um array secundário com os resultados obtidos, podendo, depois, executar a filtragem.

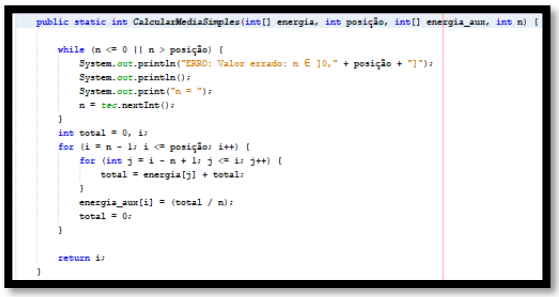

Fig. 6 – Cálculo da Média Móvel Simples

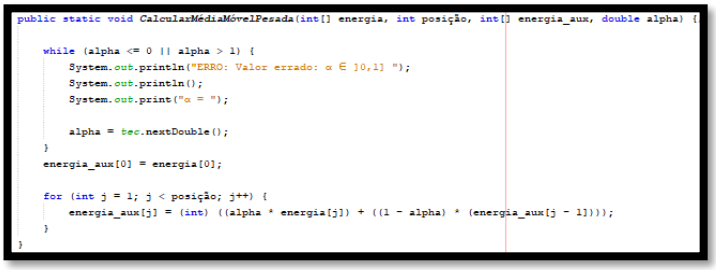

Fig. 7 – Cálculo da Média Móvel Exponencial Pesada

4.5-Ordenação de arrays:

No decorrer do projeto, foi-nos solicitado que testássemos três tipos de métodos de ordenações de arrays, para que no final conseguíssemos concluir qual seria o mais benéfico e eficaz para a totalidade do mesmo.

4.5.1-Métodos de ordenação de arrays:

- Bubble Sort:

Este mecanismo de ordenação consiste em trocar a posição de dois elementos consecutivos de um mesmo grupo, caso estes se encontrem na ordem errada.

O mecanismo acaba assim que todos os elementos estiverem ordenados da forma pretendida.

Para isso utilizamos dois outros módulos. Estes fornecem ao utilizador uma escolha de ordenar os arrays de forma crescente ou decrescente.

Segue-se um dos exemplos:

```
public static void CrescenteBubbleSort(int[] energia_aux, int posição) {

    for (int idx1 = 0; idx1 < posição; idx1++) {
        for (int idx2 = 0; idx2 < posição; idx2++) {
            if (energia_aux[idx1] < energia_aux[idx2]) {

                int auxiliar = energia_aux[idx2];
                energia_aux[idx2] = energia_aux[idx1];
                energia_aux[idx1] = auxiliar;
            }
        }
    }
}
```

Fig. 8– Exemplo de ordenação Bubble Sort (Crescente)

- Insertion Sort:

No mecanismo de Insertion Sort, são percorridas todas as posições do array, e cada um dos elementos é comparado com todos os anteriores, até que este esteja completamente ordenado.

Assim como o sistema de Bubble Sort, o Insertion Sort também disponibiliza um critério de escolha ao utilizador da forma como pretende organizar o array (de forma crescente ou decrescente).

```
public static void DecrescenteInsertionSort(int[] energia_aux, int posição) {
    int i, aux, j;
    for (i = 1; i < posição; i++) {
        aux = energia_aux[i];
        j = i - 1;

        while (j >= 0 && energia_aux[j] > aux) {
            energia_aux[j + 1] = energia_aux[j];
            j = j - 1;
        }
        energia_aux[j + 1] = aux;
    }
}
```

Fig. 9– Exemplo de ordenação Insertion Sort (Decrescente)

-Merge Sort:

Através do mecanismo de Merge Sort, o array é consecutivamente dividido em metades ordenando cada uma delas e agregando tudo no final, obtendo assim um array ordenado.

```
public static void CrescenteMergeSort(int energia[], int start, int middle, int end) {

    int temp[] = new int[end - start + 1];

    int i = start, j = middle + 1, k = 0;

    while (i <= middle && j <= end) {
        if (energia[i] <= energia[j]) {
            temp[k] = energia[i];
            k += 1;
            i += 1;
        } else {
            temp[k] = energia[j];
            k += 1;
            j += 1;
        }
    }

    while (i <= middle) {
        temp[k] = energia[i];
        k += 1;
        i += 1;
    }

    while (j <= end) {
        temp[k] = energia[j];
        k += 1;
        j += 1;
    }

    for (i = start; i <= end; i += 1) {
        energia[i] = temp[i - start];
    }

}
```

Fig. 10– Exemplo de ordenação Merge Sort (Crescente)

4.6-Teste e escolha do método mais eficaz de ordenação:

No âmbito de testar que método escolher, foi-nos disponibilizado um módulo com um algoritmo previamente preparado.

O algoritmo de testes consiste em armazenar o conteúdo da hora antes de executar a ordenação e verificar a diferença com a hora após a execução. A diferença dada é o tempo de execução do mecanismo de ordenação.

```
//TempoDeExecução----------------------------------------------------------------------------------------
    public static void TempoDeExecução(int[] energia, int posição) throws InterruptedException {

        long lStartTime = System.currentTimeMillis();

        calculation(energia, posição);

        long lEndTime = System.currentTimeMillis();

        long output = lEndTime - lStartTime;

        System.out.println("Elapsed time in milliseconds: " + output);

    }

//TempoDeExecução----------------------------------------------------------------------------------------
    private static void calculation(int[] energia, int posição) throws InterruptedException {

        CrescenteMergeSort(energia, 0, (posição - 1) / 2, posição - 1);
        TimeUnit.SECONDS.sleep(2);

    }
```

Fig. 11 - Exemplo do módulo do Tempo de Execução.

Após serem realizados todos os testes aos diferentes tipos de ordenação, foi possível concluirmos que o mais eficaz seria o Merge Sort, com um tempo de execução de: 0.0233213 segundos, sendo por sua vez o mais baixam dos três.

4.7-Codificação do sistema de previsões:

-Verificações das previsões:

Na expectativa de calcular as previsões, foi necessário realizar inúmeras validações, tendo em conta o tipo de dados introduzidos pelo utilizador.

```
public static int PrevisaoValidações(String[] datasHoras_aux, int qtd, String ParteDoDia, int n, String data_aux) throws FileNotFoundException {

    int ano, mês, dia, comparação;

    if (ParteDoDia.equalsIgnoreCase("")) {
        System.out.print("Que dia (1-31) quer prever? ");
        dia = tec.nextInt();
    } else {
        System.out.print("De que dia (1-31) quer prever a " + ParteDoDia + " ? ");
        dia = tec.nextInt();
    }
    while (dia < 1 || dia > 31) {
        System.out.println("ERRO: Dia Inválido!");
        System.out.println();
        System.out.print("Insira outro dia: ");
        dia = tec.nextInt();
    }
    System.out.println();
    System.out.print("De que mês (1-12) ? ");
    mês = tec.nextInt();
    while (mês < 1 || mês > 12) {
        System.out.println("ERRO: Mês Inválido!");
        System.out.println();
        System.out.print("Insira outro mês: ");
        mês = tec.nextInt();
    }

    comparação = -1;
    while (comparação == -1) {
        for (int i = 0; i < qtd; i++) {
            String data = datasHoras_aux[i];
            String[] dataComparação = data.split("-");
            int mesComparação = Integer.parseInt(dataComparação[1]);
            int diaComparação = Integer.parseInt(dataComparação[2]);
            if (mesComparação == mês && diaComparação == dia) {
                comparação = i;
            }
```

(…)

Fig. 12 - Exemplo de algumas validações realizadas para as previsões.

-Cálculo de previsões:

O cálculo da previsão é semelhante ao cálculo da filtragem, no entanto esta possui certas alterações nas fórmulas, de maneira a que já não seja tido em conta o valor real de energia para a mesma data.

Bibliotecas utilizadas:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import com.panayotis.gnuplot.JavaPlot;
import com.panayotis.gnuplot.plot.AbstractPlot;
import com.panayotis.gnuplot.plot.DataSetPlot;
import com.panayotis.gnuplot.style.NamedPlotColor;
import com.panayotis.gnuplot.style.PlotStyle;
import com.panayotis.gnuplot.style.Style;
import com.panayotis.gnuplot.terminal.FileTerminal;
import com.panayotis.gnuplot.terminal.GNUPlotTerminal;
import java.io.PrintWriter;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.concurrent.TimeUnit;
```

Fig. 13 – Lista de Bibliotecas Utilizadas

-Scanner:

O Scanner foi usado para realizar a leitura dos dados referentes aos ficheiros fornecidos.

-PrintWriter:

Usamos a biblioteca *PrintWriter* para fazer a transformação dos dados e imagens obtidos em ficheiros, nomeadamente nos ficheiros de output de gráficos no formato CSV e PNG.

-JavaPlot:

O JavaPlot é uma biblioteca que requere a entrada de um outro software denominado GNUplot.

Assim, esta é responsável por converter os dados e ficheiros em gráficos.

Com este comando, é-nos também permitido alterar as cores e funcionalidades dos gráficos através de certos comandos.

```
JavaPlot p = new JavaPlot();
PlotStyle myPlotStyle = new PlotStyle();
myPlotStyle.setStyle(Style.BOXES);
myPlotStyle.setLineWidth(1);
myPlotStyle.setLineType(NamedPlotColor.BLUE);
myPlotStyle.setPointType(7);
myPlotStyle.setPointSize(1);

int tab[][] = new int[7][1];

tab[1][0] = qtd1;
tab[2][0] = qtd2;
tab[3][0] = qtd3;

DataSetPlot s = new DataSetPlot(tab);
s.setTitle("Consumo de Energia");
s.setPlotStyle(myPlotStyle);
```

Fig. 14 – Exemplo de alguns comandos executados pelo JavaPlot.

5. Resultados

5.1 - Análise de Séries:

Manhãs:

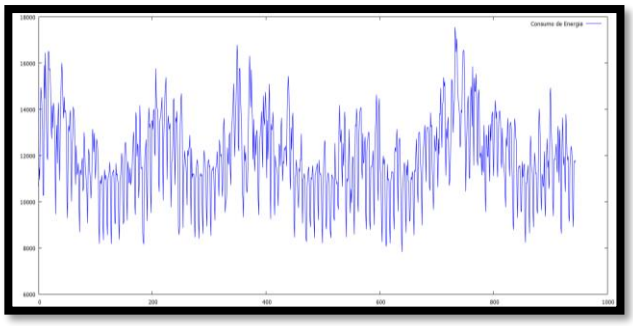

Fig. 15 – Análise de Séries (Manhã)

Anual:

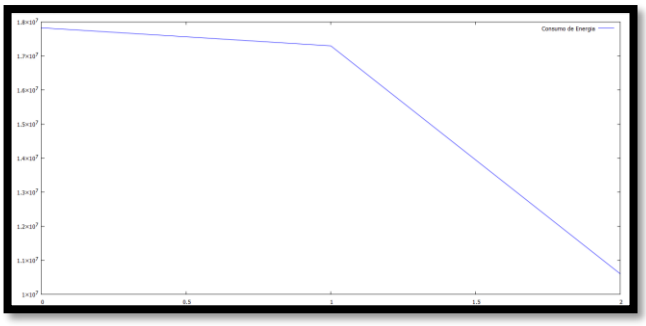

Fig. 16– Análise de Séries (Anual)

5.1.1 – Análise dos Resultados

Na figura 15 e 16, podemos observar que existem tantos pontos quanto a resolução escolhida pelo utilizador. Por exemplo, ao escolher a opção anual aparece três pontos no gráfico (fig. 13), ou seja, três anos.

5.2 - Análise de Observações:

Noite:

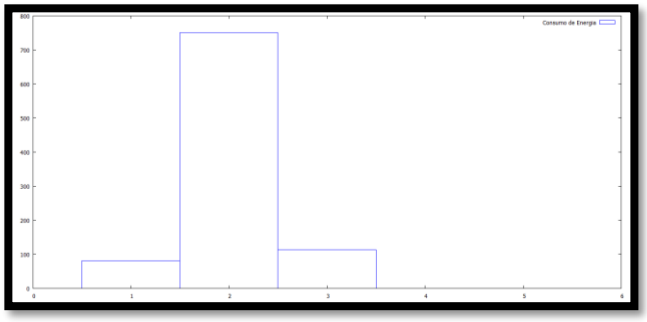

Fig. 17– Gráfico da Análise de Observações (Noite)

```
Média global: 12999 MW
```

Fig. 19 – Print da linha de comandos (Média global (Noite))

Diário:

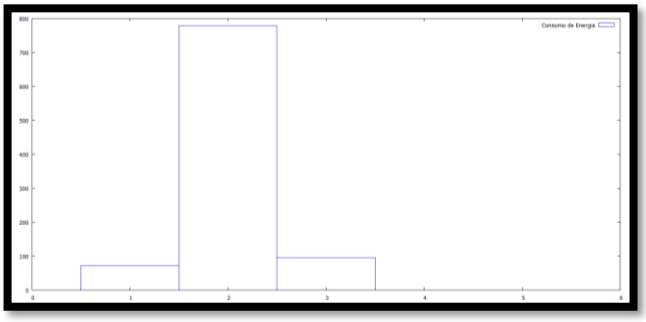

Fig.18 – Gráfico da Análise de Observações (Diário)

```
Média global: 48334 MW
```

Fig.20 – Print da linha de comandos (Média global (Diário))

5.2.1 – Análise dos Resultados

Na figura 17e 18, é de esperar que a coluna do centro inclua um maior número de observações pelo facto de estar num intervalo entre menos vinte porcento (-20%) da média e mais vinte porcento (+20%) da média.

5.3 - Ordenar Valores:

Tarde (ordem decrescente):

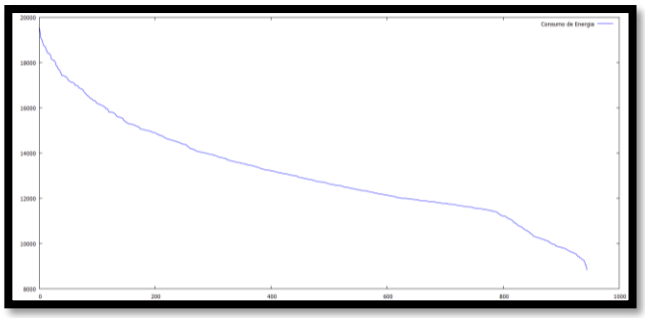

Fig.21 – Ordenar Valores-Decrescente (Tarde)

Mensal (ordem crescente):

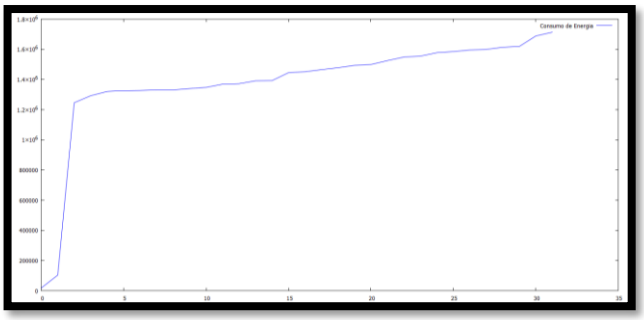

Fig. 22 – Ordenar Valores-Crescente (Mensal)

5.3.1 – Análise dos Resultados

Na figura 21 e 22, os valores apresentam-se pela ordem inserida pelo utilizador (relativamente ao eixo das ordenadas) como é pretendido.

5.4 - Filtragens:

Manhã (média móvel simples):

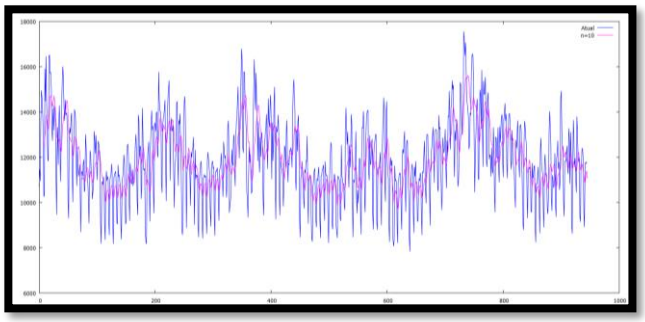

Fig. 23 – Filtragem-MMS (Manhã)

Manhã (média móvel pesada):

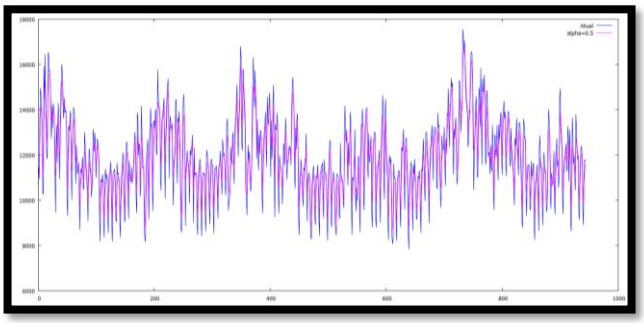

Fig. 24 – Filtragem-MMP (Manhã)

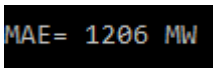

Fig. 25 – Print (Erro médio absoluto-MMS (Manhã))

Fig. 26– Print (Erro médio absoluto-MMP (Manhã))

5.4.1 – Análise dos Resultados

Na figura 23 e 24, é notório uma suavização do gráfico atual sendo que a suavização utilizando a MMS é maior que a MMP neste caso apresentado, devido aos valores de n (10) e de α (0,5). O erro da filtragem é maior quanto maior for a suavização.

5.5 - Previsão:

Anual (próximo ano, 2019):

```
ERRO: O último ano da Série Temporal não se encontra completo!
```

Fig. 27 – Print (Previsão (Anual))

Madrugada (dia 2/12/2017 para n=10):

```
A previsao é de 10178 MW
```

Fig. 28 – Print (Previsão-MMS (Madrugada))

Madrugada (dia 2/12/2017 para α=0,5):

```
A previsao é de 10165 MW
```

Fig. 29 – Print (Previsão-MMP (Madrugada))

5.5.1 – Análise dos Resultados

Na figura 27, aparece uma imagem de "ERRO" pelo facto de o ficheiro dado (DAYTON.csv) não conter o ano de 2018 completo!

Na figura 28 e 29, é calculada uma previsão através da Média Móvel Simples e da Média Móvel Pesada, respetivamente, que apresentam um valor aproximado ao da madrugada desse mesmo dia (valor da série original -> 10567 MW).

6. Conclusão

Ao longo do Projeto utilizamos a metodologia de EduScrum, e dessa forma conseguimos que todas as tarefas do trabalho fossem divididas em subtarefas atribuídas a diferentes elementos do grupo. Com isto conseguimos uma melhor dinâmica de grupo e também uma melhor cooperação entre todos os elementos.

Apesar de todas as dificuldades passadas com a elaboração de certas partes do trabalho, o grupo conseguiu superar todas essas adversidades uma vez que estávamos todos muito unidos e cada um era responsável pela sua parte, sempre que terminada a sua tarefa mostrava-se preparado para ajudar noutra tarefa mostrando assim o empenho de todos no trabalho.

Com a realização deste trabalho percebemos a importância das séries temporais no estudo e compreensão do comércio atual, uma vez que estas permitem prever futuros comportamentos de determinada ação, por exemplo. E com isso ajuda os investidores e os próprios comerciante a perceberem o futuro do seu negócio e assim tomar as melhores decisões para o futuro do seu negócio.

No código desenvolvido é de realçar o uso do conceito da modularização que permite a divisão do código em módulos mais pequenos, como por exemplo diferentes métodos de ordenação e as diferentes funcionalidades implementadas. No código realizado foi implementada a biblioteca JavaPlot que permite a demonstração de dados em gráficos, aprimorando a experiência de utilizador com o programa.

Quando confrontado com um projeto de maior dificuldade, o grupo manteve a sua postura bem como empenhava-se ainda mais para que essa dificuldade desaparecesse o mais rapidamente possível e para que o grupo pudesse dessa maneira avançar no projeto.

É de referir que uma das nossas maiores dificuldades foi mesmo a realização dos testes unitários, sendo que a tivemos de ultrapassar mesmo no final do projeto. Tirando esse ponto de maior dificuldade o resto do trabalho conseguimos obter os resultados esperados e realizar tudo o que era suposto embora pudéssemos gerir melhor o tempo mais para o final do projeto.

Referências

Barata, A. (17 de Dezembro de 2019). *MoodleISEP*. Obtido de ISEP: https://moodle.isep.ipp.pt/course/view.php?id=7688

Panayotis, K. (5 de Março de 2015). *SOURCEFORGE*. Obtido de JavaPlot: https://sourceforge.net/projects/gnujavaplot/

Williams, T., & Kelley, C. (Março de 2010). Obtido de gnuplot 5.0: An interactive plotting program.: http://gnuplot.sourceforge.net/

ANEXOS

ANEXO A _ Testes Unitários

```
import org.junit.Assert;
import org.junit.Test;
import static org.junit.Assert.*;
import projecto_lapr1_1dlk.Projecto_LAPR1_1DLK;

public class ProjetoLapr1Test {

    public ProjetoLapr1Test() {
    }



    @Test
    public void test_CrescenteMergeSort(){
       //arrange
        int [] testeordenar = new int [10];
        int [] testeordenar_final = new int [10];
        int start=0, middle=0, end=10;


        testeordenar[0]= 3;
        testeordenar[1]= 10;
        testeordenar[2]= 500;
        testeordenar[3]= 1000;
        testeordenar[4]= 42;
        testeordenar[5]= 82;
        testeordenar[6]= 123;
        testeordenar[7]= 0;
        testeordenar[8]= 9872;
        testeordenar[9]= 0;
        //-------------------------------------------------
        testeordenar_final[0]= 0;
        testeordenar_final[1]= 0;
        testeordenar_final[2]= 3;
        testeordenar_final[3]= 10;
        testeordenar_final[4]= 42;
        testeordenar_final[5]= 82;
        testeordenar_final[6]= 123;
        testeordenar_final[7]= 500;
        testeordenar_final[8]= 1000;
        testeordenar_final[9]= 9872;

      //act
         int result []= Projecto_LAPR1_1DLK.CrescenteMergeSortTest(testeordenar, start, middle, end);
      //assert
        Assert.assertArrayEquals( result, testeordenar_final);
```