Higgs bosons production analysis

Filippo Imboccioli, filippo.imboccioli@studio.unibo.it

Abstract

The dataset analyzed in this project is: HIGGS.
It was published along with this paper: Searching for Exotic Particles in High-Energy Physics with Deep Learning.
The project consisit in the application of **3** between **classification and clustering algorithms**:

- Decision Tree Classifier
- Random Forest Classifier
- K-Means Clustering

to the original dataset and some meaningful variations of it:

- original dataset (with dropped NaNs)
- dataset with only features selected by the variance
- dataset with only function columns
- dataset with only kinematic features columns

The experiment rely on **PySpark** and **HDFS**.

1 Architecture

The **cluster** is made up of **3 Vagrant** VMs. Each VM is created with its own specific Vagrantfile that specifies its ip address. The HADOOP cluster architecture is made up of:

- **node-master**: *NameNode-SecondaryNameNode-ResourceManager*

```ruby
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "ubuntu/jammy64"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # NOTE: This will enable public access to the opened port
  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine and only allow access
  # via 127.0.0.1 to disable public access
  # config.vm.network "forwarded_port", guest: 8888, host: 8888, host_ip: "127.0.0.1"

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  config.vm.network "private_network", ip: "10.0.0.3"

  # Create a public network, which generally matched to bridged network.
  # Bridged networks make the machine appear as another physical device on
  # your network.
  # config.vm.network "public_network"

  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
  # config.vm.synced_folder "../data", "/vagrant_data"

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant. These expose provider-specific options.
  # Example for VirtualBox:
  #
  config.vm.provider "virtualbox" do |vb|
  #   # Display the VirtualBox GUI when booting the machine
  #   vb.gui = true
  #
  #   # Customize the amount of memory on the VM:
    vb.memory = "4096"
    vb.customize ["modifyvm", :id, "--vram", "32"]
  end
  #
  # View the documentation for the provider you are using for more
  # information on available options.

  # Enable provisioning with a shell script. Additional provisioners such as
  # Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
  # documentation for more information about their specific syntax and use.
  # config.vm.provision "shell", inline: <<-SHELL
  #   apt-get update
  #   apt-get install -y apache2
  # SHELL
  # config.vm.provision :shell, path: "bootstrap.sh"
end
```

- **node1:** *DataNode-NodeManager*

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "ubuntu/jammy64"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # NOTE: This will enable public access to the opened port
  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine and only allow access
  # via 127.0.0.1 to disable public access
  # config.vm.network "forwarded_port", guest: 8888, host: 8888, host_ip: "127.0.0.1"

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  config.vm.network "private_network", ip: "10.0.0.4"
```

```ruby

  # Create a public network, which generally matched to bridged network.
  # Bridged networks make the machine appear as another physical device on
  # your network.
  # config.vm.network "public_network"

  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
  # config.vm.synced_folder "../data", "/vagrant_data"

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant. These expose provider-specific options.
  # Example for VirtualBox:
  #
  config.vm.provider "virtualbox" do |vb|
  #   # Display the VirtualBox GUI when booting the machine
  #   vb.gui = true
  #
  #   # Customize the amount of memory on the VM:
    vb.memory = "4096"
    vb.customize ["modifyvm", :id, "--vram", "32"]
  end
  #
  # View the documentation for the provider you are using for more
  # information on available options.

  # Enable provisioning with a shell script. Additional provisioners such as
  # Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
  # documentation for more information about their specific syntax and use.
  # config.vm.provision "shell", inline: <<-SHELL
  #   apt-get update
  #   apt-get install -y apache2
  # SHELL
  # config.vm.provision :shell, path: "bootstrap.sh"
end
```

- **node2:** *DataNode-NodeManager*

```ruby
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.
  config.vm.box = "ubuntu/jammy64"

  # Disable automatic box update checking. If you disable this, then
  # boxes will only be checked for updates when the user runs
  # `vagrant box outdated`. This is not recommended.
  # config.vm.box_check_update = false

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine. In the example below,
  # accessing "localhost:8080" will access port 80 on the guest machine.
  # NOTE: This will enable public access to the opened port
  # config.vm.network "forwarded_port", guest: 80, host: 8080

  # Create a forwarded port mapping which allows access to a specific port
  # within the machine from a port on the host machine and only allow access
  # via 127.0.0.1 to disable public access
  # config.vm.network "forwarded_port", guest: 8888, host: 8888, host_ip: "127.0.0.1"

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  config.vm.network "private_network", ip: "10.0.0.5"

  # Create a public network, which generally matched to bridged network.
  # Bridged networks make the machine appear as another physical device on
  # your network.
  # config.vm.network "public_network"

  # Share an additional folder to the guest VM. The first argument is
  # the path on the host to the actual folder. The second argument is
  # the path on the guest to mount the folder. And the optional third
  # argument is a set of non-required options.
  # config.vm.synced_folder "../data", "/vagrant_data"

  # Provider-specific configuration so you can fine-tune various
  # backing providers for Vagrant. These expose provider-specific options.
  # Example for VirtualBox:
  #
  config.vm.provider "virtualbox" do |vb|
  #   # Display the VirtualBox GUI when booting the machine
  #   vb.gui = true
  #
  #   # Customize the amount of memory on the VM:
    vb.memory = "4096"
    vb.customize ["modifyvm", :id, "--vram", "32"]
  end
  #
  # View the documentation for the provider you are using for more
  # information on available options.

  # Enable provisioning with a shell script. Additional provisioners such as
  # Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
  # documentation for more information about their specific syntax and use.
  # config.vm.provision "shell", inline: <<-SHELL
  #   apt-get update
  #   apt-get install -y apache2
  # SHELL
  # config.vm.provision :shell, path: "bootstrap.sh"
end
```

1.1 Configure the hadoop cluster

For each VM

1. **Configuration of the system:** a new user is created: *hadoop* user. Then it is added to the sudoers group.

```
sudo adduser hadoop
sudo visudo #for adding hadoop in sudoers
```

```
hadoop   ALL=(ALL:ALL) ALL #add to the sodoers
```

```
su - hadoop
mkdir .ssh
```

2. **Creation of the host file on each node:** the */etc/hosts* file is updated to let the VMs know each other replacing the raw ip with a name.

```
sudo nano /etc/hosts
```

```
#add the following rows:
10.0.0.3    node-master
10.0.0.4    node1
10.0.0.5    node2
```

3. **Distribution of authentication key-pairs for the Hadoop user:** SSH authentication key-pairs are distributed for the hadoop user. The *node-master* has to generate a public key that has to be added to its *authorized_keys* and the other nodes' ones as well.

   (a) Login to node-master as the Hadoop user, and generate an SSH key:

```
ssh-keygen -b 4096
```

(b) In each VM, make a new file master.pub in the /home/hadoop/.ssh directory. Paste the public key into this file and save the changes.

```
cd .ssh
cat id_rsa.pub
nano master.pub
```

(c) Copy the key file into the authorized key store

```
cat ~/.ssh/master.pub >> ~/.ssh/authorized_keys
```

4. **Downloading and unpacking hadoop binaries:** the hadoop binaries are downloaded and unpacked

   (a) Log into node-master as the hadoop user, download the Hadoop tarball from Hadoop project page, and unzip it:

```
cd
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.4/hadoop-3.3.4.tar.gz
tar -xzf hadoop-3.3.4.tar.gz
mv hadoop-3.3.4 hadoop
```

5. **Installing Java:** Java is installed and the *JAVA_HOME* is added to the *PATH* and to the *ENVIRONMENT VARIABLES*. On all VMs:

   (a) Installing Java:

```
sudo apt-get update
sudo apt-get install openjdk-8-jdk
```

   (b) Setting JAVA_HOME

      i. Edit ~/hadoop/etc/hadoop/hadoop-env.sh and replace this line:

```
export JAVA_HOME=${JAVA_HOME}
```

      ii. This has to be done for all the VMs and for the following file: [/etc/profile, .bashrc, .profile]

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre
export PATH=${PATH}:${JAVA_HOME}

export HADOOP_HOME=/home/hadoop/hadoop
export PATH=${PATH}:${HADOOP_HOME}/bin:${HADOOP_HOME}/sbin
```

6. **Setting NameNode location:** the *NameNode* location is set in the *core-site.xml* configuration file:

   (a) Update ~/hadoop/etc/hadoop/core-site.xml file to set the NameNode location to node-master on port 9000:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://node-master:9000</value>
    </property>
</configuration>
```

7. **Setting the path for HDFS:** the path for HDFS is set in the *hdfs-site.xml* configuration file

   (a) Edit ~/hadoop/etc/hadoop/hdfs-site.xml to resemble the following configuration:

```
<configuration>
    <property>
            <name>dfs.namenode.name.dir</name>
            <value>/home/hadoop/data/nameNode</value>
    </property>

    <property>
            <name>dfs.datanode.data.dir</name>
            <value>/home/hadoop/data/dataNode</value>
    </property>

    <property>
            <name>dfs.replication</name>
            <value>1</value>
    </property>
</configuration>
```

8. **Setting and configuring YARN as job scheduler:** YARN job scheduler is set and configured in *yarn-site.xml*.

   (a) Edit the ∼/hadoop/etc/hadoop/mapred-site.xml file, setting YARN as the default framework for MapReduce operations:

```
<configuration>
    <property>
            <name>mapreduce.framework.name</name>
            <value>yarn</value>
    </property>
    <property>
            <name>yarn.app.mapreduce.am.env</name>
            <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
    </property>
    <property>
            <name>mapreduce.map.env</name>
            <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
    </property>
    <property>
            <name>mapreduce.reduce.env</name>
            <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
    </property>
</configuration>
```

   (b) Edit ∼/hadoop/etc/hadoop/yarn-site.xml, which contains the configuration options for YARN

```
<configuration>
    <property>
            <name>yarn.acl.enable</name>
            <value>0</value>
    </property>

    <property>
            <name>yarn.resourcemanager.hostname</name>
            <value>node-master</value>
    </property>

    <property>
            <name>yarn.nodemanager.aux-services</name>
            <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

9. **Configuring workers:** the *workers* file is used by startup to start required daemons on all nodes. For this purpose this file is edited to include both the nodes. Edit ∼/hadoop/etc/hadoop/workers to include both of the nodes:

```
node1
node2
```

10. **Configuring memory allocation:** the memory allocation properties are configured in *yarn-site.xml*

   (a) Edit ∼/hadoop/etc/hadoop/yarn-site.xml and add the following lines:

```
<property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>1536</value>
</property>

<property>
        <name>yarn.scheduler.maximum-allocation-mb</name>
        <value>1536</value>
</property>
```

```

<property>
        <name>yarn.scheduler.minimum-allocation-mb</name>
        <value>128</value>
</property>

<property>
        <name>yarn.nodemanager.vmem-check-enabled</name>
        <value>false</value>
</property>
```

11. **Configuration of map-reduce**: the configuration for map reduce is set in the *mapred-site.xml*

    (a) Edit ~/hadoop/etc/hadoop/mapred-site.xml and add the following lines:

    ```
    <property>
            <name>yarn.app.mapreduce.am.resource.mb</name>
            <value>512</value>
    </property>

    <property>
            <name>mapreduce.map.memory.mb</name>
            <value>256</value>
    </property>

    <property>
            <name>mapreduce.reduce.memory.mb</name>
            <value>256</value>
    </property>
    ```

12. **Duplicating configuration files on each node**: it is necessary to duplicate configuration files on each node

    (a) Copy the Hadoop binaries to worker nodes:

    ```
     cd /home/hadoop/
     scp hadoop-*.tar.gz node1:/home/hadoop
     scp hadoop-*.tar.gz node2:/home/hadoop
    ```

    (b) Unzip the binaries, rename the directory:

    ```
    tar -xzf hadoop-3.3.4.tar.gz
    mv hadoop-3.3.4 hadoop
    ```

    (c) Run on the master to copy the Hadoop configuration files to the worker nodes:

    ```
    for node in node1 node2; do
    scp ~/hadoop/etc/hadoop/* $node:/home/hadoop/hadoop/etc/hadoop/;
    done
    ```

13. **Formatting HDFS**: on the *node-master*:

    (a) HDFS needs to be formatted like any classical file system. For this purpose, this command has to be run on the *node-master*.

    ```
    hdfs namenode -format
    ```

    (b) The HDFS has to be started and stopped with

    ```
    start-dfs.sh
    ```

    and

    ```
    stop-dfs.sh
    ```

    (c) It is necessary to run the command

    ```
    start-yarn.sh
    ```

    to start a *ResourceManager* on *node-master* and *NodeManager* on *node1* and *node2*

    (d) Check that every process is running with the jps command on each node. On node-master, you should see the following (the PID number will be different)

    ```
    jps
    ```

The expected output on *node-master* is something like that

```
xxxxx Jps
yyyyy NameNode
zzzzz SecondaryNameNode
iiiii ResourceManager
```

The expected output on **data nodes** is something like that:

```
jjjjj Jps
kkkkk DataNode
lllll NodeManager
```

14. **Monitoring YARN and Hadoop**:

    - This has to be run to monitor hadoop:

    ```
    hdfs dfsadmin -report
    hdfs dfsadmin -help
    http://localhost:9870 after vagrant ssh -- -L 9870:localhost:9870
    ```

    - This has to be run to minitor yarn:

    ```
    hdfs dfsadmin -report
    hdfs dfsadmin -help
    http://localhost:9870 after vagrant ssh -- -L 9870:localhost:9870

    yarn node -list
    yarn application -list
    http://localhost:8088
    ```

15. **File loading**: for the file loading step, it is necessary to run on the *node-master*:

```
hdfs dfs -mkdir -p /user/hadoop
hdfs dfs -mkdir dataset
hdfs dfs -put PAMAP2_Dataset dataset
```

1.2 Setting up and configuring Spark on top of the Hadoop YARN cluster

1. **Downloading and installing Spark Binaries**: the Spark binaries are downloaded and unpacked. Log on node-master as the hadoop user, and run:

```
cd /home/hadoop
wget https://dlcdn.apache.org/spark/spark-3.4.0/spark-3.4.0-bin-hadoop3.tgz
tar -xvf spark-3.4.0-bin-hadoop3.tgz
rm spark-3.4.0-bin-hadoop3.tgz
mv spark-3.4.0-bin-hadoop3 spark
```

2. **Adding the Spark binaries directory to PATH**: Edit /home/hadoop/.profile and add the following line:

```
PATH=/home/hadoop/spark/bin:$PATH
```

3. **Integration of Spark with YARN**: to communicate with the YARN Resource Manager, Spark needs to be aware of your Hadoop configuration. This is done via the HADOOP_CONF_DIR environment variable. The SPARK_HOME variable is not mandatory, but is useful when submitting Spark jobs from the command line.

   (a) Edit the hadoop user profile /home/hadoop/.profile and add the following lines:

   ```
   export HADOOP_CONF_DIR=/home/hadoop/hadoop/etc/hadoop
   export SPARK_HOME=/home/hadoop/spark
   export LD_LIBRARY_PATH=/home/hadoop/hadoop/lib/native:$LD_LIBRARY_PATH
   ```

   (b) Restart the session by logging out and logging in again.

   (c) Rename the spark default template config file:

   ```
   mv $SPARK_HOME/conf/spark-defaults.conf.template $SPARK_HOME/conf/spark-
   defaults.conf
   ```

   (d) Edit $SPARK_HOME/conf/spark-defaults.conf and set spark.master to yarn:

   ```
   spark.master    yarn
   ```

4. **Configuring memory allocation**: it is important to give YARN Containers the maximum allowed memory. If the memory requested is above the maximum allowed, YARN will reject creation of the container, and the Spark application won't start. Get the value of yarn.scheduler.maximum-allocation-mb in $HADOOP_CONF_DIR/yarn-site.xml. This is the maximum allowed value, in MB, for a single container. Make sure that values for Spark memory allocation, configured in the following section, are below the maximum.

5. **Configuring the Spark driver memory allocation in cluster mode**: In order to configure the Spark Driver Memory Allocation in cluster mode, it is necessary to edit the file $SPARK_HOME/conf/spark-defaults.conf:

   ```
   spark.driver.memory    512m
   ```

   In cluster mode, the Spark Driver runs inside YARN Application Master. The amount of memory requested by Spark at initialization is configured either in spark-defaults.conf, or through the command line.

6. **Configuring the Spark executors' memory allocation**: the Spark Executors' memory allocation is calculated based on two parameters inside $SPARK_HOME/conf/spark-defaults.conf:

   - spark.executor.memory: sets the base memory used in calculation
   - spark.yarn.executor.memoryOverhead: is added to the base memory. It defaults to 7% of base memory, with a minimum of 384MB

   To set executor memory to 512MB, edit $SPARK_HOME/conf/spark-defaults.conf and add the following line:

   ```
   spark.executor.memory          512m
   ```

2 Data description

The dataset is composed of:

- Number of patterns: 100000
- Number of features: 28

All the features columns are float values, while the class label has 2 classes, also those data has been produced using Monte Carlo simulations.

The first column is the binary class label:

- **Higgs bosons production**
  - **1**: signal process that produced Higgs bosons
  - **0**: background process that did *NOT* produced Higgs bosons

The first 21 features (columns 2-22) are **kinematic properties** measured by the particle detectors in the accelerator:

- **lepton pT**: The transverse momentum of a lepton (electron or muon)
- **lepton eta**: The pseudorapidity of a lepton, which describes its angle relative to the beam axis
- **lepton phi**: The azimuthal angle of a lepton
- **missing energy magnitude**: The magnitude of the missing energy, which represents the undetected particles in the event
- **missing energy phi**: The azimuthal angle of the missing energy
- **jet 1 pt**: The transverse momentum of the first jet
- **jet 1 eta**: The pseudorapidity of the first jet
- **jet 1 phi**: The azimuthal angle of the first jet
- **jet 1 b-tag**: A b-tagging value for the first jet, which indicates the likelihood of it being a bottom quark jet
- **jet 2 pt**: The transverse momentum of the second jet
- **jet 2 eta**: The pseudorapidity of the second jet
- **jet 2 phi**: The azimuthal angle of the second jet
- **jet 2 b-tag**: A b-tagging value for the second jet, which indicates the likelihood of it being a bottom quark jet
- **jet 3 pt**: The transverse momentum of the third jet
- **jet 3 eta**: The pseudorapidity of the third jet
- **jet 3 phi**: The azimuthal angle of the third jet
- **jet 3 b-tag**: A b-tagging value for the third jet, which indicates the likelihood of it being a bottom quark jet
- **jet 4 pt**: The transverse momentum of the fourth jet
- **jet 4 eta**: The pseudorapidity of the fourth jet
- **jet 4 phi**: The azimuthal angle of the fourth jet
- **jet 4 b-tag**: A b-tagging value for the fourth jet, which indicates the likelihood of it being a bottom quark jet

The last 7 features are **functions** of the first 21 features: these are high-level features derived by physicists to help discriminate between the two classes:

- **m_jj**: The invariant mass of the two leading jets
- **m_jjj**: The invariant mass of the three leading jets
- **m_lv**: The invariant mass of the lepton and missing energy system
- **m_jlv**: The invariant mass of the lepton, missing energy, and leading jet system
- **m_bb**: The invariant mass of the two b-tagged jets
- **m_wbb**: The invariant mass of the lepton, missing energy, and two leading b-tagged jets system
- **m_wwbb**: The invariant mass of the two W bosons and two leading b-tagged jets system

The dataset is divided in:

- **train**: 70% -> 70.000 rows
- **test**: 30% -> 30.000 rows

3 Workflows

The dataset is analyzed and does **NOT** contain any *Nan* or *null* value.
For every one of the 4 experiments:

- **Original features**
- **Feature selection**
- **Dropping the 21 kinematic properties columns (LOW-LEVEL features)**
- **Dropping the 7 functions columns (HIGH-LEVEL features)**

the dataset was untouched (non-normalized) and normalized.
Train and Test results of the **3 algorithms** will follow.

3.1 Original features

1. **Creation of train-test sets:** assembly and normalise or not the features
2. **Train results**:

accuracy/f1-score	**classification tree**	**random forest**	**k-means**
non-normalised	1.0/1.0	1.0/1.0	0.5010/0.4899
normalised	1.0/1.0	1.0/1.0	0.5632/0.5129

3. **Test results**:

accuracy/f1-score	**classification tree**	**random forest**	**k-means**
non-normalised	1.0/1.0	1.0/1.0	0.4912/0.4800
normalised	1.0/1.0	1.0/1.0	0.5651/0.5159

3.2 Feature selection

This method is used for the selection of the columns that have a variance greater than a certain threshold. If the variance of a column is very low then it will have a negligible role for the discrimination. In this case, the higher the variance, the better the opportunity to discriminate between different classes. It applies:

1. **Feature selection:** features with a variance greater than threshold are selected
2. **Creation** of **train-test** datasets: assembly and normalise or not the features
3. **Train results**:

accuracy/f1-score	**classification tree**	**random forest**	**k-means**
non-normalised	1.0/1.0	1.0/1.0	0.5038/0.4926
normalised	1.0/1.0	1.0/1.0	0.4978/0.4982

4. **Test results**:

accuracy/f1-score	**classification tree**	**random forest**	**k-means**
non-normalised	1.0/1.0	1.0/1.0	0.4977/0.4865
normalised	1.0/1.0	1.0/1.0	0.4935/0.4940

3.3 Dropping the 21 kinematic properties columns (LOW-LEVEL features)

The decision to test this procedure is supported by the following compelling reasons:

- **Domain Knowledge**: Physicists have derived the 7 function columns specifically to aid in distinguishing between signal and background processes. By testing on these features alone, you are focusing on the knowledge that experts have deemed important, potentially leading to better performance.
- **Dimensionality Reduction**: High-level features are created by combining the low-level features, potentially capturing essential patterns and relationships. Using only the function columns can help reduce dimensionality and eliminate noise from individual low-level features.
- **Feature Engineering Evaluation**: Testing on the function columns allows you to assess the effectiveness of the feature engineering process. It helps validate whether the derived high-level features are indeed informative for the classification task.
- **Comparison to Prior Work**: If previous research or experiments have already established the importance of these high-level features, testing solely on them allows you to compare your model's performance directly to prior work.

1. **Dropping column**
2. **Creation** of **train-test datasets**: assembly and normalise or not the features
3. **Train results**:

accuracy/f1-score	**classification tree**	**random forest**	**k-means**
non-normalised	1.0/1.0	1.0/1.0	0.9999/0.9999
normalised	1.0/1.0	1.0/1.0	0.5581/0.4976

4. **Test results**:

accuracy/f1-score	**classification tree**	**random forest**	**k-means**
non-normalised	1.0/1.0	1.0/1.0	1.0/1.0
normalised	1.0/1.0	1.0/1.0	0.5596/0.4990

3.4 Dropping the 7 functions columns (HIGH-LEVEL features)

The decision to test this procedure is supported by the following compelling reasons:

- **Interpretability**: The kinematic properties are direct measurements from particle detectors, and they have a clear physical interpretation. By testing on these features alone, the model's predictions can be more interpretable and explainable.
- **Feature Importance**: By evaluating the model's performance on the kinematic properties only, you can determine the intrinsic importance of these low-level features in distinguishing between signal and background processes. This can provide insights into which features have the most discriminative power.
- **Complexity Reduction**: Using only the kinematic properties simplifies the model and reduces the number of features, which can be beneficial if the high-level features are not contributing significantly to the classification task.
- **Baseline Performance**: Testing on the kinematic properties allows you to establish a baseline performance. It helps to understand how well the model performs when only the raw detector measurements are used, and no higher-level domain knowledge is incorporated.

1. **Dropping column**
2. **Creation** of **train-test datasets**: assembly and normalise or not the features
3. **Train results**:

accuracy/f1-score	**classification tree**	**random forest**	**k-means**
non-normalised	1.0/1.0	1.0/1.0	0.5010/0.4899
normalised	1.0/1.0	1.0/1.0	0.4994/0.4997

4. **Test results**:

accuracy/f1-score	**classification tree**	**random forest**	**k-means**
non-normalised	1.0/1.0	1.0/1.0	0.4911/0.4799
normalised	1.0/1.0	1.0/1.0	0.5041/0.5046

3.5 Values prediction

This method was *NOT* implemented because the dataset do not contain any *NaN* or *null* values. However, to avoid loosing too many rows in case of missing values, they can be predicted with:

- **Nearest neighbour**: looks at the nearest non-NaN value
- **Linear interpolation**: fitting and interpolating a line between the nearest left and right non NaN values

4 Final considerations and results

4.1 General classification methods comparison

As it is possible to notice, the clustering **k-means** algorithm performs much worse than the **classification tree** and **random forest** due to the fact that **k-means** relies directly on the raw data. In general, k-means is not the most suitable algorithm for binary classification tasks like distinguishing between signal and background processes, as it is primarily designed for unsupervised clustering. With it there is **no-learning**, only **memorization**. The main drawback of kNN is that it is a "lazy" algorithm, all computations is postponed to test time. If it is good for modelling complex data distributions, because it is a non-parametric instance-based algorithm, on the other hand it is only as good as the features and the distance in feature space are. For this reason it can suffer from the **curse of dimensionality**. In this case 28 features are used, and applying euclidean distance in such a huge dimensional space, will often result in bad results.
On the contrary **classification tree** algorithm performs **perfectly** due to the fact that it looks for the feature that allows it to discriminate between classes in a better way. With the help of entropy or gini impurity indices, it can find the feature that allows it to reduce the cahos in labels' distribution.
Along with it the other classification algorithm that performs **perfectly** is **random forest** that uses ensemble techniques that are able to provide boostings in performance due to the fact that it can rely on more classifiers at the same time.

4.2 Pre-processing methods comparison

Both the classification tree and random forest models achieve **perfect accuracy (1.0) and F1-score (1.0)** on both the training and test sets when using all features, so we need to look at the K-mean to draw some interesting conclusions.

- **Original Features**: The k-means clustering algorithm performs relatively poorly with an accuracy of around 50% and an F1-score below 50%. Normalizing the features generally improves the performance of all three models, especially for the k-means algorithm.

- **Feature Selection**: Feature selection based on variance does not seem to have a significant impact on the performance of the classification tree and random forest models. For k-means, feature selection results in decreased accuracy and F1-score, indicating that dropping features with low variance might not be beneficial for this clustering algorithm.

- **Dropping 21 Kinematic Properties (Low-level Features)**: When dropping the 21 kinematic properties columns and using only the 7 high-level function columns, the classification tree and random forest models achieve perfect accuracy and F1-score on both training and test sets. Interestingly, **k-means achieves almost perfect accuracy and F1-score on the non-normalized dataset**. When the dataset is not normalized, each feature has its original scale and range. K-means is a distance-based clustering algorithm, and the distances between data points are sensitive to the scale of the features. **If some features have a much larger scale than others, they can dominate the distance calculations, leading to uneven cluster assignments.** This scenario might have resulted in clusters that are well-separated in the feature space, leading to better k-means performance. This could be due to the high-level features capturing more meaningful patterns that lead to distinct clusters. In contrast, when the dataset is normalized, all features are scaled to a similar range, so this process aims to prevent certain features from dominating the clustering algorithm due to their larger scale. This normalized dataset might not have provided enough discriminatory information for k-means to form well-separated clusters. The high-level features alone might not be sufficient to accurately distinguish between signal and background processes, leading to more overlapping clusters.
  In essence, normalization changes the data's scale and range, potentially altering the relative importance of the features for clustering. In the non-normalized case, **dropping the kinematic properties might have accidentally removed noise or irrelevant information, leading to well-defined clusters.** However, in the normalized case, the

data distribution could be different, making the kinematic properties important for clustering. K-means is sensitive to the initial placement of centroids and can converge to different solutions for different initializations, therefore, the discrete results in the normalized dataset might be due to k-means converging to different local optima, which can vary based on the initial centroid positions.

- **Dropping Function Columns (High-level Features)**: For k-means, dropping the high-level features leads to a slight decrease in accuracy and F1-score compared to using all features.

Overall Observations:

- The classification tree and random forest models perform remarkably well on the classification task using the given features, achieving perfect accuracy and F1-score. Normalizing the features generally does not improve improve the performance, as it scales the features and prevents some from dominating the learning process.
- The k-means clustering algorithm, even when using the correct number of clusters (k=2 for signal and background), does not perform as well as the classification models, indicating that the data's inherent patterns are not easily separable using k-means.
- Dropping the 21 kinematic properties (low-level features) results in nearly perfect performance for all models, suggesting that the high-level features are more informative for distinguishing between signal and background processes.
- Dropping the 7 function columns (high-level features) does not significantly impact the classification tree and random forest models' performance, indicating that the low-level features are already highly discriminative. For k-means, this does not lead to substantial performance changes.