The usage of Deep Learning for QCD background estimation

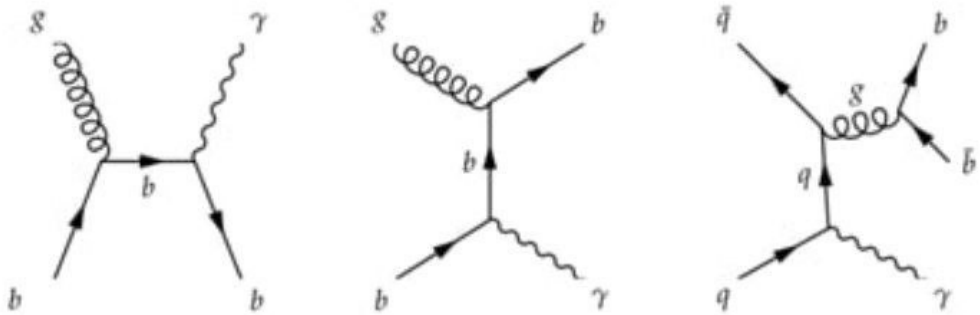
Andrii Len
Taras Shevchenko National University of Kyiv

IRIS-HEP Summer Fellowship
Mentor: Ece Asilar (CERN)

October 19, 2022

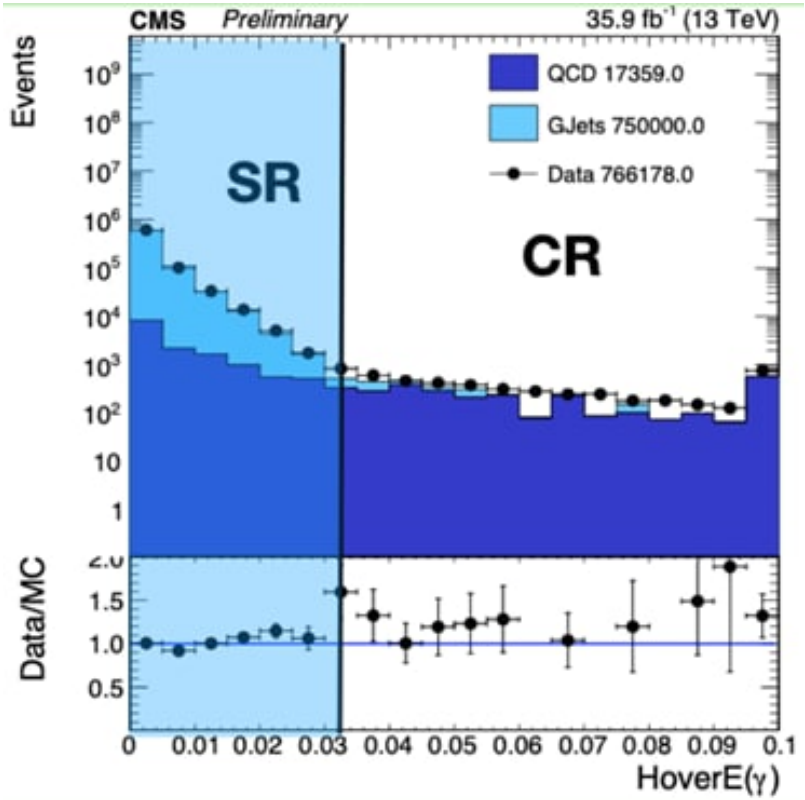
Introduction and motivation

- $\gamma + b$ jets @ 13 TeV with full Run II which has about 137 fb^{-1} luminosity
- Direct/Prompt photons: photons produced directly from the hard scattering of partons



- $\gamma + b$ jets analysis provides:
  - Testing ground for pQCD
  - Information about the PDF of b quarks and gluons
  - Enlightening of the dynamics of hard scattering process by angular correlations between γ and jets
  - Testing ground for different Monte Carlo generators

Data selections for DNN training and testing



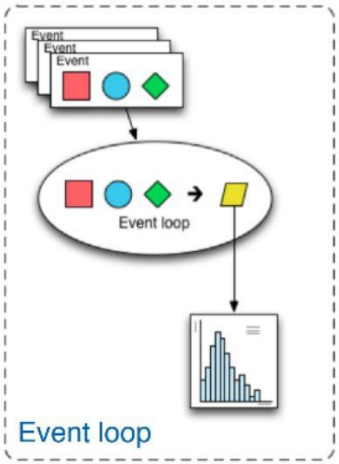
HoverE cuts for training: For QCD Background HoverE>0.04 and for Gjets signal HoverE<0.04. For model's testing both Signal and Background are taken from the Signal Region (HoverE<0.04)

Columnar Analysis: a Paradigm Shift

Moving from **Event loop** analysis to **Columnar** analysis

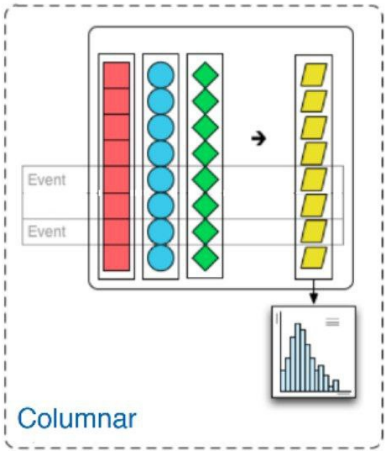
Event loop analysis:

- Load relevant values for a specific event into local variables
- Evaluate several expressions
- Store derived values
- Repeat (explicit outer loop)

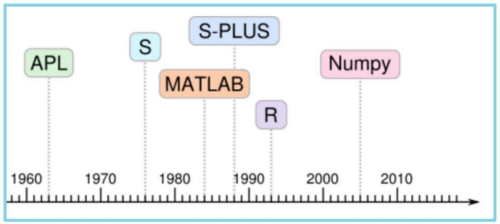


Columnar analysis:

- Load relevant values for many events into contiguous arrays
- Evaluate several **array programming** expressions (implicit inner loops)
- Store derived values



Array programming:



Simple operations to act on an entire array at once

Coffea Processor

```
from coffea import hist, processor

class MyProcessor(processor.ProcessorABC):
    def __init__(self, flag=False):
        self._flag = flag
        self._accumulator = processor.dict_accumulator({
            # Define histograms
        })

    @property
    def accumulator(self):
        return self._accumulators

    def process(self, df):
        output = self.accumulator.identity()

        # PHYSICS GOES HERE

        return output

    def postprocess(self, accumulator):
        return accumulator

p = MyProcessor()
```

Implementing cuts and reading files

```
# setting cuts on our events
goodevents = events[(events.ngoodPhoton==1)&(events.ngoodbJet>=1)&(events.ngoodPhoton_hoe>0.04)
```

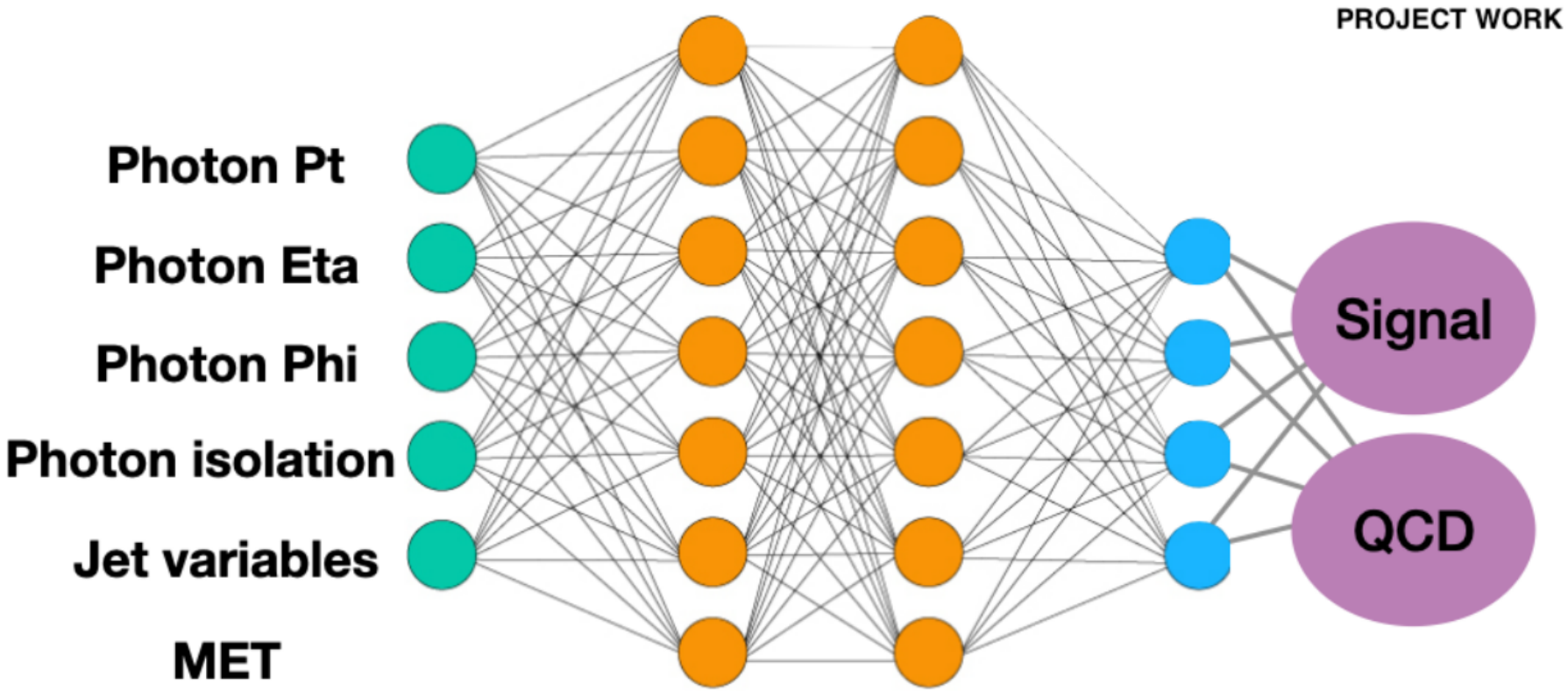
```
# zipping only needed variables
Photons = ak.zip(
    {
        "pt" :goodevents.goodPhoton_pt,
        "eta" :goodevents.goodPhoton_eta,
        "phi" :goodevents.goodPhoton_phi,
        "sieie" :goodevents.goodPhoton_sieie,
        "hoe" :goodevents.goodPhoton_hoe,
    }
    behavior=candidate.behavior,
)
```

```
# defining root files to process
fileset_GJEt_nocuts = "/eos/user/a/alen/G1JetSource/allGJet_nocutsmini.json"
```

```
# setting processor executor, schema and output
futures_run = processor.Runner(
    executor = processor.FuturesExecutor(compression=0, workers=8),
    schema=BaseSchema,
    maxchunks=10000,
)

out_gjet = futures_run(
    fileset_GJEt_nocuts,
    "Events",
    processor_instance=MyProcessor(fileset_GJEt_nocuts)
)
```

Our preliminary DNN model

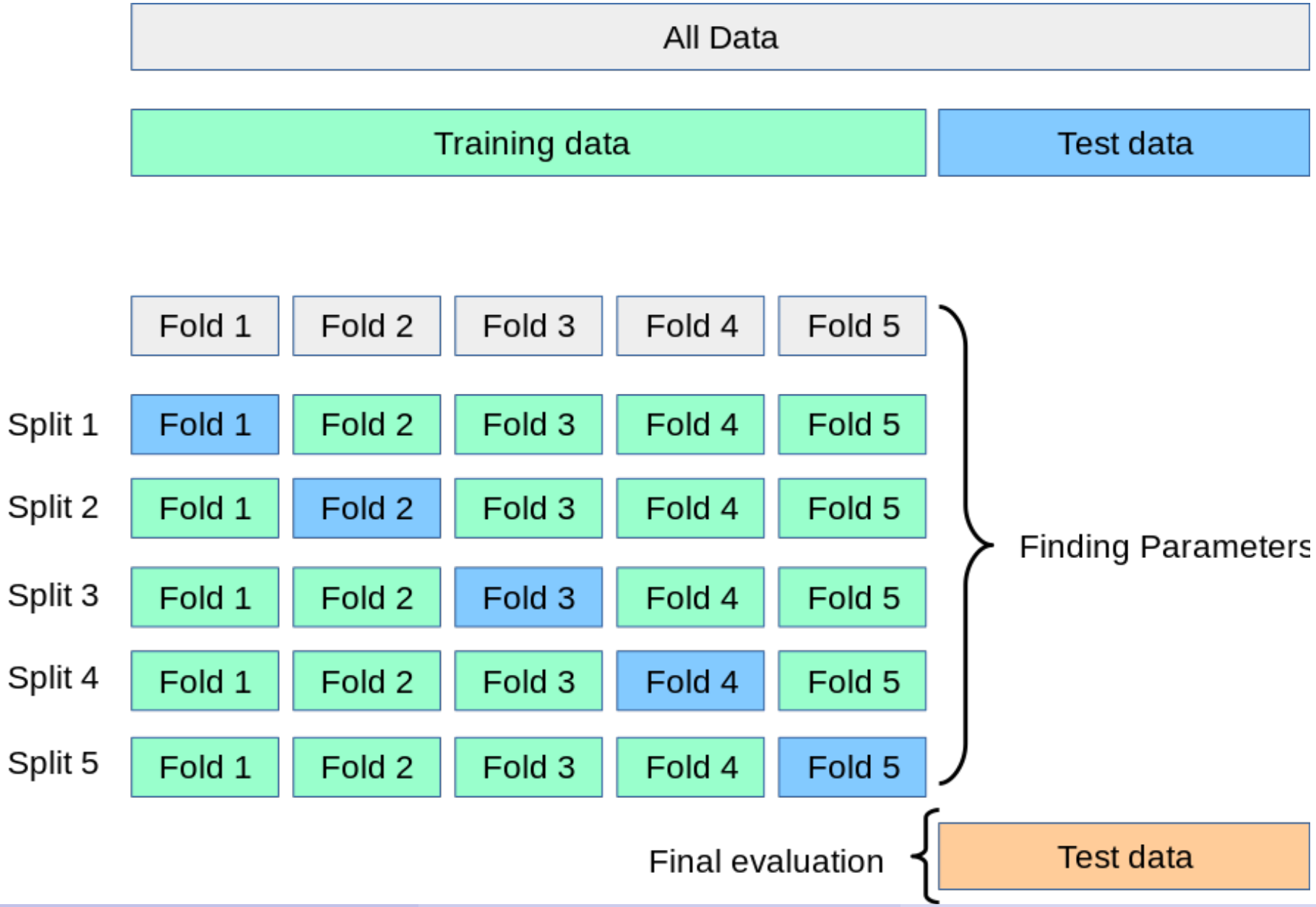


Hyperparameters

•Optimizing algorithm : SGD, Adadelta, Adam, Adamax, Nadam, Adagrad, RMSprop
•Learning rate : 0.001, 0.002, 0.1, 0.3, 0.5, 0.9
•Momentum : 0.001, 0.002, 0.1, 0.3, 0.5, 0.9
•Weight initialisation mode : uniform, normal, lecun uniform, normal, zero, glorot normal, glorot uniform, he normal, he uniform
• Number of deep layers : 50, 30, 20, 15, 10, 8, 6, 4, 3, 2, 1
• Number of neurons per layer : different combinations from 800 to 10
• Number of epochs : 50 epochs with an early stop condition i.e the training is stopped when the performance reaches a plateau
• Input Variables : Different sets of input variables are tested

(Click here to go to keras)

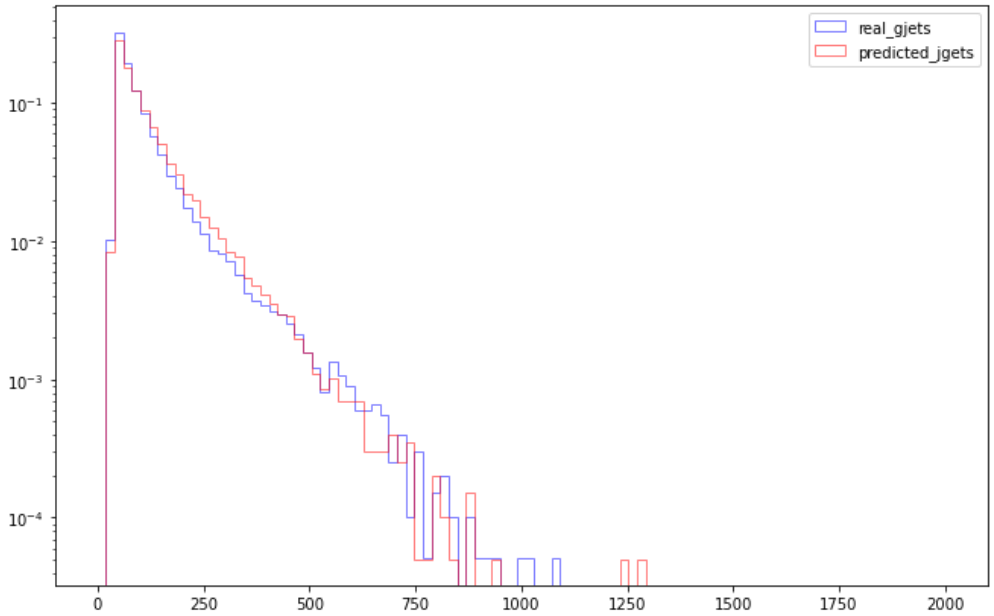
K-fold cross-validation



Hyperparameter search results

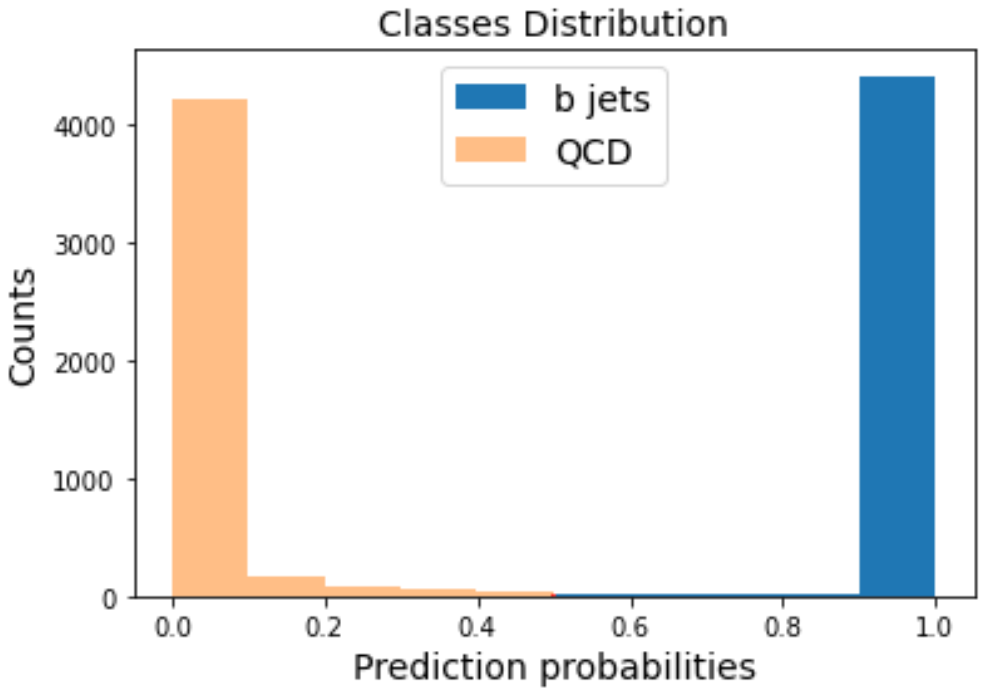
- Optimizing algorithm : Adamax
- Learning rate : 0.002
- Weight initialisation mode : normal
- Number of deep layers : 3
- Number of neurons per layer : 200, 110, 20
- Number of epochs : 20
- Input Variables : Photons, jets and bjets variables

Model's performance



Real and predicted Photon pt distribution from the Monte Carlo Signal Region comparison

Model's performance



Results of training and testing model with QCD from real data and Signal from MC:

- AUC score : 99.72%
- Sensitivity : 97.71%
- Specificity : 98.86%

Conclusions

- All simulated and real data for 3 years was collected using coffea
- Regions of this data needed for Deep Learning were established and cuts were defined
- Was implemented DNN and were found it's best Hyperparameters
- Method with Monte Carlo events was tested and gave results consistent with the expectations
- Running our metod with Real Data

Thank you for your attention!!!