# JLEE DSC180B Project Demo

January 26, 2022

[1]:
```python
import uproot
import numpy as np
import os
import random
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import altair as alt
from collections import Counter

%matplotlib inline
```

[36]:
```python
def path_generator(t:str, eda=True) -> list:
    '''
    Approximately (size of 50 QCD dataset) == (size of 14 Hbb dataset)
    '''
    lst = []

    if t.upper() == 'QCD':
        main = '/home/h8lee/teams/DSC180A_FA21_A00/a11/train_mass_qcd/\
QCD_HT{low}to{high}_TuneCP5_13TeV-madgraph-pythia8/'
        if eda:
            num_data = 10

        bounds = [
            [1000,1500],
            [1500,2000],
            [2000, 'Inf'],
            [500,700],
            [700,1000]
        ]

        for bound in bounds:
            low, high = bound
            fp = main.format(low=low, high=high)
            all_files = os.listdir(fp)
            samples = random.sample(all_files, k=num_data)
```

```
            # There's this one hidden file under (700-1000) bound
            while '.nano_mc2017_174_Skim.root.ViGCYO' in samples:
                samples = random.sample(all_files, k=num_data) # Re-sample

            files = [os.path.join(fp, sample) for sample in samples]
            lst += files # In total, randomly generate filepaths to 50␣
↪different QCD .root files
    elif t.upper() == 'SIGNAL':
        main = '/home/h8lee/teams/DSC180A_FA21_A00/a11/train_mass_hbb/\
BulkGravitonToHHTo4Q_MX-600to6000_MH-15to250_part{}_TuneCP5_13TeV-madgraph_pythia8/
↪'
        if eda:
            num_data = 4

        parts = [1,2]

        for part in parts:
            # Since files in Hbb directory1 are smaller than those in Hbb␣
↪directory2,
            # sample more from directory1 to balance size of samples generating␣
↪from
            # directory2
            # (11 .root files in dir1) == (3 .root files in dir2)

#             if part==1:
#                 num_data = 11
#             else:
#                 num_data = 3

            fp = main.format(part)
            all_files = os.listdir(fp)
            samples = random.sample(all_files, k=num_data)
            files = [os.path.join(fp, sample) for sample in samples]

            lst += files
    return lst
```

[37]:
```
qcd_eda_sets = path_generator('QCD', eda=True)
signal_eda_sets = path_generator('signal', eda=True)
```

[38]:
```
def load_jet_features(fps):
    '''
    For all files at defined filepaths,
    extract jet features from each of them as well as their type
    '''
    jet_features = []
```

```
    unnecesssary_attrs = [
        'fj_idx',
        'fj_genRes_mass',
        'fj_lsf3'
    ]
    df = pd.DataFrame()

    for i in range(len(fps)):
        path = fps[i]
        f = uproot.open(path)
        tree = f['Events']

        if i==0:
            attrs = [branch.name for branch in tree.branches]
            jet_features += list(filter(lambda x:x.startswith('fj'), attrs))
            jet_features = [feat for feat in jet_features if feat not in␣
↪unnecesssary_attrs] # drop sterile attributes

        features = tree.arrays(jet_features, library='np')
        df = pd.concat([df, pd.DataFrame(features)], axis=0)

    df = df.reset_index(drop=True)

    return df
```

---

**Validating the labels of QCD/signal jet samples**

[39]:
```
df_qcd = load_jet_features(qcd_eda_sets)
display(df_qcd.head())
print('\n', f'{df_qcd.shape[0]} randomly generated QCD jet samples')
```

```
   fj_pt    fj_eta    fj_phi   fj_mass  fj_msoftdrop  fj_deepTagMD_H4qvsQCD  \
0  572.5  0.799072  0.082703  127.8750     85.125000                -1000.0
1  347.0 -0.325134 -2.830566   84.8125      3.986328                -1000.0
2  578.0 -0.938354  0.759888  133.7500      4.753906                -1000.0
3  315.0 -2.022949 -2.620117  215.5000    222.250000                -1000.0
4  528.0  0.015053 -1.083008   59.1250      1.471680                -1000.0

   fj_deepTag_HvsQCD  fj_PN_H4qvsQCD  fj_PN_XbbvsQCD  fj_genjetmsd  ...  \
0            -1000.0        0.117758        0.000385     83.062500  ...
1            -1000.0        0.000004        0.000995      5.769531  ...
2            -1000.0        0.000013        0.003182     12.640625  ...
3            -1000.0        0.000773        0.000216    292.500000  ...
4            -1000.0        0.000069        0.002145      3.216797  ...

   fj_genW_decay  fj_genWstar_decay  fj_evt_met_covxx  fj_evt_met_covxy  \
```

```
0          -99.0              -99.0            1604.0               265.0
1          -99.0              -99.0            1604.0               265.0
2          -99.0              -99.0            2176.0              1028.0
3          -99.0              -99.0            2176.0              1028.0
4          -99.0              -99.0            1002.0              -610.0

   fj_evt_met_covyy  fj_evt_met_dphi  fj_evt_met_pt  fj_evt_met_sig  \
0             822.0         2.690247      46.284767        1.890625
1             822.0        -0.679670      46.284767        1.890625
2            2640.0        -3.082642     219.364868       14.039062
3            2640.0         0.297363     219.364868       14.039062
4            1880.0        -2.609357      17.122852        0.212280

   fj_evt_pupmet_pt  fj_evt_pupmet_dphi
0         44.854141            2.516418
1         44.854141           -0.853498
2        205.903931           -3.108032
3        205.903931            0.271973
4         22.418184           -2.543439

[5 rows x 57 columns]


 371331 randomly generated QCD jet samples
```

[40]:
```python
df_signal = load_jet_features(signal_eda_sets)
display(df_signal.head())
print('\n', f'{df_signal.shape[0]} randomly generated signal jet samples')
```

```
    fj_pt    fj_eta    fj_phi  fj_mass  fj_msoftdrop  fj_deepTagMD_H4qvsQCD  \
0  430.50 -1.148926 -1.613525  344.750    342.000000                -1000.0
1  430.50 -0.531250  1.475342  126.750     12.109375                -1000.0
2  409.25  1.776855 -0.478271  171.125    167.000000                -1000.0
3  382.25  1.072266  2.770996  157.125    151.000000                -1000.0
4  587.00 -0.079727 -0.807129  466.750    474.000000                -1000.0

   fj_deepTag_HvsQCD  fj_PN_H4qvsQCD  fj_PN_XbbvsQCD  fj_genjetmsd  ...  \
0            -1000.0        0.423500        0.921231    302.250000  ...
1            -1000.0        0.002387        0.557630     21.734375  ...
2            -1000.0        0.004376        0.929233    155.500000  ...
3            -1000.0        0.007784        0.003990    159.625000  ...
4            -1000.0        0.040538        0.001870    477.500000  ...

   fj_genW_decay  fj_genWstar_decay  fj_evt_met_covxx  fj_evt_met_covxy  \
0          -99.0              -99.0             632.0             86.75
1          -99.0              -99.0             632.0             86.75
2          -99.0              -99.0            2552.0           -606.00
```

```
3           -99.0              -99.0          2552.0           -606.00
4           -99.0              -99.0          2728.0           -594.00

   fj_evt_met_covyy  fj_evt_met_dphi  fj_evt_met_pt  fj_evt_met_sig  \
0            1892.0        -1.440186      30.616550        1.472656
1            1892.0         1.754133      30.616550        1.472656
2            1200.0        -3.110578      65.651611        1.567383
3            1200.0        -0.076660      65.651611        1.567383
4            2208.0        -0.319824      53.534805        1.097656

   fj_evt_pupmet_pt  fj_evt_pupmet_dphi
0         30.935398           -0.855713
1         30.935398            2.338605
2         44.169460            3.101318
3         44.169460           -0.147949
4         56.205227           -0.374023

[5 rows x 57 columns]
```

```
 487040 randomly generated signal jet samples
```

**Data validation** Validate the type of jets in our samples; each jet should only be associated to one unique type of QCD/signal

[41]:
```python
# QCD
# For this checkup, we only need label attribute

IS_QCDb = 'fj_isQCDb'
IS_QCDothers = 'fj_isQCDothers'
all_attrs = df_qcd.columns.tolist()
start_idx = all_attrs.index(IS_QCDb)
end_idx = all_attrs.index(IS_QCDothers)+1

qcd_labels = all_attrs[start_idx:end_idx]
```

[42]:
```python
df_qcd_labels = df_qcd[qcd_labels]
display(df_qcd_labels.head())
```

```
   fj_isQCDb  fj_isQCDbb  fj_isQCDc  fj_isQCDcc  fj_isQCDlep  fj_isQCDothers
0          0           0          0           0            0               1
1          0           0          1           0            0               0
2          0           0          0           0            0               1
3          0           0          0           0            0               1
4          0           0          0           0            1               0
```

[43]:
```python
# We want each jet corresponding to exactly one type

print(f'Each jet corresponds to exactly one type:\
 {len(df_qcd_labels.sum(axis=1).unique()) == 1}')
```

```
Each jet corresponds to exactly one type: True
```

[44]:
```python
# How many jets are there for different QCD types?

display(df_qcd_labels.sum(axis=0).sort_values(ascending=False).
 ↪to_frame(name='Count'))
```

| | Count |
|---|---|
| fj_isQCDothers | 225829 |
| fj_isQCDlep | 80905 |
| fj_isQCDcc | 26058 |
| fj_isQCDc | 25787 |
| fj_isQCDb | 6543 |
| fj_isQCDbb | 6209 |

[45]:
```python
# Signal jets
# For this checkup, we only need label attribute

IS_HBB = 'fj_H_bb'
IS_HQQ = 'fj_H_qq'
all_attrs = df_signal.columns.tolist()
start_idx = all_attrs.index(IS_HBB)
end_idx = all_attrs.index(IS_HQQ)+1

signal_labels = all_attrs[start_idx:end_idx]
```

[46]:
```python
df_signal_labels = df_signal[signal_labels]

# We're only going to include signal jets
# of types H_bb, H_cc, H_qq for performing EDA
df_signal_labels = df_signal_labels[
    (df_signal_labels['fj_H_bb'] == 1) |
    (df_signal_labels['fj_H_cc'] == 1) |
    (df_signal_labels['fj_H_qq'] == 1)
]

# Drop observations that are associated to more than single type
df_signal_labels['temp'] = df_signal_labels['fj_H_bb'] +␣
 ↪df_signal_labels['fj_H_cc'] + df_signal_labels['fj_H_qq']
print(f'Before filtering: {df_signal_labels.shape[0]} rows', '\n')
```

```
df_signal_labels = df_signal_labels[df_signal_labels['temp'] == 1].
 ↪drop(columns='temp')
print(f'After filtering: {df_signal_labels.shape[0]} rows')
```

```
Before filtering: 460901 rows

After filtering: 460875 rows
```

[47]:
```
# We want each jet corresponding to exactly one type

print(f'Each jet corresponds to exactly one type:\
 {len(df_signal_labels.sum(axis=1).unique()) == 1}')
```

```
Each jet corresponds to exactly one type: True
```

[48]:
```
# How many jets are there for each signal type?

display(df_signal_labels.sum(axis=0).sort_values(ascending=False).
 ↪to_frame(name='Count'))
```

| | Count |
|---|---|
| fj_H_bb | 154163 |
| fj_H_cc | 154125 |
| fj_H_qq | 152587 |

---

**EDA #1** Plot distribution of our target attribute(`fj_genjetmsd`), generator-level soft drop mass, of QCD jets according to their type

[49]:
```
# Filtering using the validation results

signal_idx = df_signal_labels.index.tolist()
df_signal = df_signal.filter(items=signal_idx, axis=0)
```

[50]:
```
# Create temporary `class` label to differentiate QCD jets from signal jets
# Then concatenate QCD dataset to signal dataset

df_qcd['Type'] = 'QCD'
df_signal['Type'] = 'Signal'

df_qcd_and_signal = pd.concat([df_qcd, df_signal], axis=0)
display(df_qcd_and_signal.head())
```

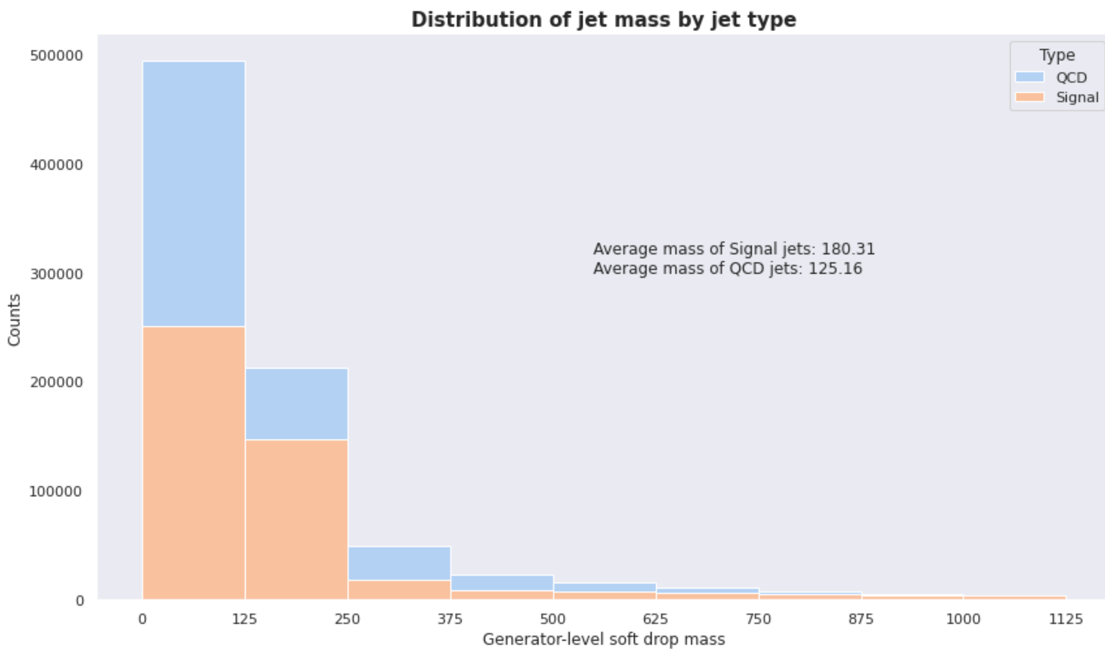
```
   fj_pt    fj_eta    fj_phi   fj_mass  fj_msoftdrop  fj_deepTagMD_H4qvsQCD  \
0  572.5  0.799072  0.082703  127.8750     85.125000                -1000.0
1  347.0 -0.325134 -2.830566   84.8125      3.986328                -1000.0
```

```
2  578.0 -0.938354  0.759888  133.7500      4.753906                 -1000.0
3  315.0 -2.022949 -2.620117  215.5000    222.250000                 -1000.0
4  528.0  0.015053 -1.083008   59.1250      1.471680                 -1000.0

   fj_deepTag_HvsQCD  fj_PN_H4qvsQCD  fj_PN_XbbvsQCD  fj_genjetmsd  ...  \
0            -1000.0        0.117758        0.000385     83.062500  ...
1            -1000.0        0.000004        0.000995      5.769531  ...
2            -1000.0        0.000013        0.003182     12.640625  ...
3            -1000.0        0.000773        0.000216    292.500000  ...
4            -1000.0        0.000069        0.002145      3.216797  ...

   fj_genWstar_decay  fj_evt_met_covxx  fj_evt_met_covxy  fj_evt_met_covyy  \
0              -99.0            1604.0             265.0             822.0
1              -99.0            1604.0             265.0             822.0
2              -99.0            2176.0            1028.0            2640.0
3              -99.0            2176.0            1028.0            2640.0
4              -99.0            1002.0            -610.0            1880.0

   fj_evt_met_dphi  fj_evt_met_pt  fj_evt_met_sig  fj_evt_pupmet_pt  \
0         2.690247      46.284767        1.890625         44.854141
1        -0.679670      46.284767        1.890625         44.854141
2        -3.082642     219.364868       14.039062        205.903931
3         0.297363     219.364868       14.039062        205.903931
4        -2.609357      17.122852        0.212280         22.418184

   fj_evt_pupmet_dphi  Type
0            2.516418   QCD
1           -0.853498   QCD
2           -3.108032   QCD
3            0.271973   QCD
4           -2.543439   QCD

[5 rows x 58 columns]
```

[51]:
```
avg_mass = df_qcd_and_signal.groupby('Type')['fj_genjetmsd'].mean()
avg_mass_qcd = round(avg_mass.loc['QCD'], 2)
avg_mass_signal = round(avg_mass.loc['Signal'], 2)

text = f'Average mass of Signal jets: {avg_mass_signal:.5}\n\
Average mass of QCD jets: {avg_mass_qcd:.5}'
```

[52]:
```
# Used `.displot()` from seaborn for visualization

_ = sns.set(context='notebook', rc={'figure.figsize':(14,8)},
            style='dark', palette='pastel')
ax = sns.histplot(x='fj_genjetmsd', data=df_qcd_and_signal, hue='Type',
                  bins=range(0, 1250, 125), multiple='stack')
```

```
_ = ax.set_title('Distribution of jet mass by jet type', fontdict={'size':15,␣
↪'weight':'bold'})
_ = ax.set_xlabel('Generator-level soft drop mass')
_ = ax.text(550, 300000, text)
_ = ax.set_xticks(range(0,1250,125))
_ = ax.set_ylabel('Counts')
```




---

[53]:
```
# BELOW COULD BE USED FOR VALIDATION OF signal jets DATASET

# Hbb jets are known to be the most common type of
# jet that contains Higgs boson
# This is because Higgs boson is more likely to decay into
# *exactly* two b-quarks in a single proton-proton event
# These b quarks are known for their relatively heavier weights
# compared to other elementary particles
# which gives them longer lifespan
# One conspicuous characteristic of Hbb jets
# is hence their long lifespan
# which is enabled by heavy weights of Higgs boson and its decay product, b␣
↪quarks
# Let's now explore if this finding holds in our datset
# using `fj_genjetmsd` attribute
```

```
# Does above trend hold for QCD jets also?
# i.e. Do certain types of QCD jet tend to be heavier than otehr types?
```

### 0.0.1 EDA #2 – QCD

Does presence of secondary vertices in a jet have any effect on jet mass?

[54]:
```
def load_num_sv(fps, jet_type='QCD'):
    '''
    For all files at defined filepaths,
    extract secondary vertex features from
    each of the file
    '''
    NUMPY = 'np'
    SV_PT_LOG = 'sv_pt_log'
    FJ_GENJETMSD = 'fj_genjetmsd'
    num_svs = []
    jet_mass = []

    for i in range(len(fps)):
        path = fps[i]
        f = uproot.open(path)
        tree = f['Events']
        sv_pt_logs = tree.arrays(SV_PT_LOG, library=NUMPY)[SV_PT_LOG]
        num_sv = list(map(lambda sublst: len(list(filter(lambda x: x != 0,␣
 ↪sublst))), sv_pt_logs))
        num_svs += num_sv

        # Jet masses(target)
        masses = tree.arrays(FJ_GENJETMSD, library=NUMPY)[FJ_GENJETMSD].tolist()
        jet_mass += masses

#     df = df.reset_index(drop=True)

    return num_svs, jet_mass
```

**QCD**

[55]:
```
qcd_num_svs, qcd_jet_mass = load_num_sv(qcd_eda_sets)

avg_qcd_num_svs = np.mean(qcd_num_svs)
avg_qcd_jet_mass = np.mean(qcd_jet_mass)

med_qcd_num_svs = np.median(qcd_num_svs)
med_qcd_jet_mass = np.median(qcd_jet_mass)

qcd_num_svs_counter = Counter(qcd_num_svs)
```

```python
temp = qcd_num_svs_counter.items()
qcd_num_svs_counts = sorted(temp, reverse=True, key=lambda x:x[1])
# Later create 2 x 2 dataframe; indexing QCD and Signal, columns of mean, median
```

[56]:
```python
df_qcd_num_svs_counts = pd.DataFrame(qcd_num_svs_counts,
                                     columns=['# of SVs in a jet', 'counts'],
                                    ).set_index('# of SVs in a jet')

display(df_qcd_num_svs_counts)
print(f'Majority of QCD jets in our data has no to only few secondary vertex\
 recorded', '\n')
```

```
                   counts
# of SVs in a jet
0                  134975
1                  120403
2                   65650
3                   29871
4                   12337
5                    5004
6                    1886
7                    1205
```

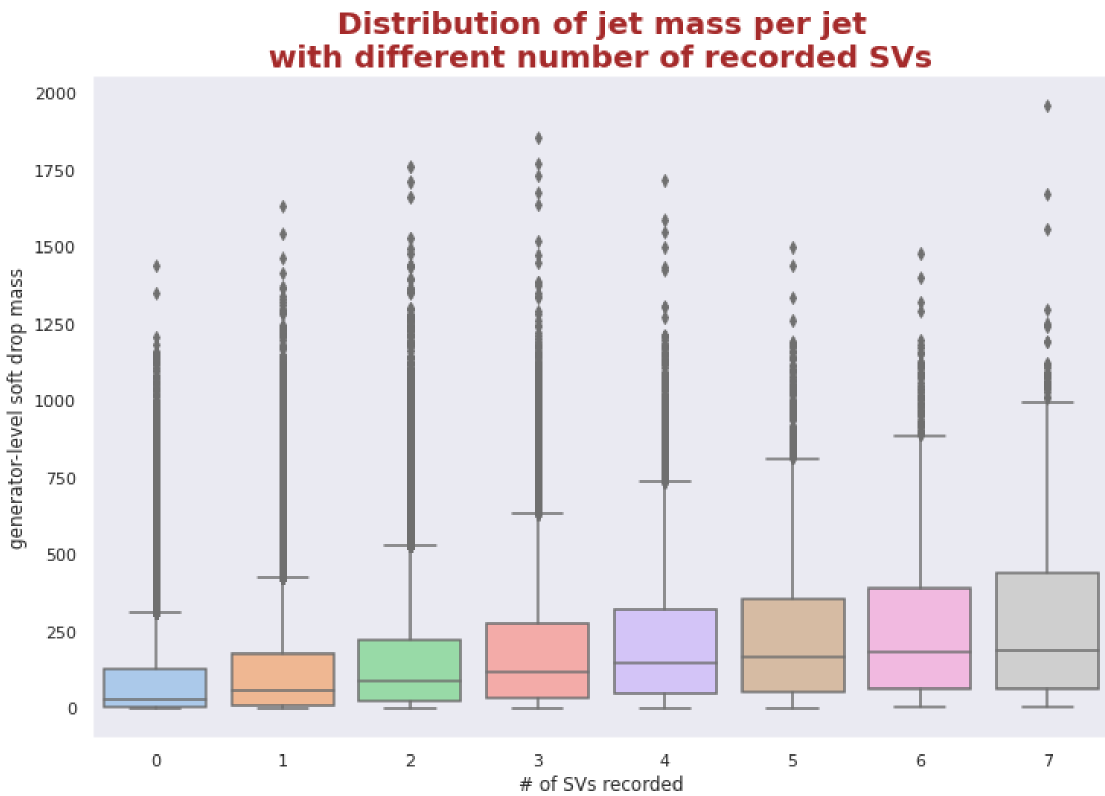
```
Majority of QCD jets in our data has no to only few secondary vertex recorded
```

[57]:
```python
qcd_dict = {
    '# of SVs recorded':qcd_num_svs,
    'generator-level soft drop mass':qcd_jet_mass
}

qcd_df = pd.DataFrame(qcd_dict)
display(qcd_df.head())
```

```
   # of SVs recorded  generator-level soft drop mass
0                  0                       83.062500
1                  1                        5.769531
2                  0                       12.640625
3                  2                      292.500000
4                  0                        3.216797
```

[58]:
```python
# Boxplot to show relationship between them

_ = sns.set(rc={'figure.figsize':(12,8)})

qcd_box = sns.boxplot(x='# of SVs recorded',
```

```
            y='generator-level soft drop mass',
            data=qcd_df, palette='pastel')

_ = qcd_box.grid(False)
TITLE = 'Distribution of jet mass per jet\nwith different number of recorded␣
 ↪SVs'
_ = qcd_box.set_title(TITLE, fontdict={'size':20, 'weight':'bold', 'color':
 ↪'brown'})
```




---

**Signals**

[59]:
```
signal_num_svs, signal_jet_mass = load_num_sv(signal_eda_sets)

avg_signal_num_svs = np.mean(signal_num_svs)
avg_signal_jet_mass = np.mean(signal_jet_mass)

med_signal_num_svs = np.median(signal_num_svs)
med_signal_jet_mass = np.median(signal_jet_mass)

signal_num_svs_counter = Counter(signal_num_svs)
```

```
temp = signal_num_svs_counter.items()
signal_num_svs_counts = sorted(temp, reverse=True, key=lambda x:x[1])
```

[60]:
```
df_signal_num_svs_counts = pd.DataFrame(signal_num_svs_counts,
                                       columns=['# of SVs in a jet', 'counts']
                                       ).set_index('# of SVs in a jet')

display(df_signal_num_svs_counts)
print(f'Unlike QCD, majority of Signal jets in our data has at least 1␣
 ↪secondary vertices\
 recorded')
```

```
                   counts
# of SVs in a jet
1                  134771
2                  111269
0                   96257
3                   70523
4                   38671
5                   19256
6                    9045
7                    7248
```

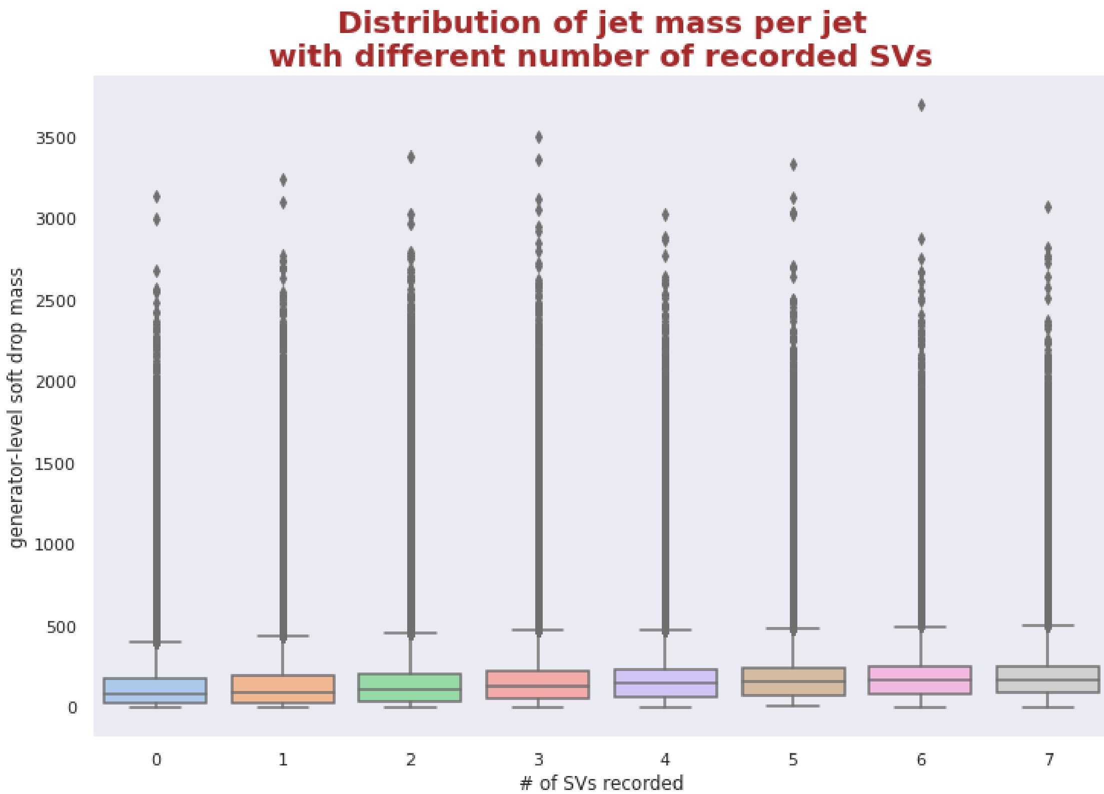
Unlike QCD, majority of Signal jets in our data has at least 1 secondary vertices recorded

[61]:
```
signal_dict = {
    '# of SVs recorded':signal_num_svs,
    'generator-level soft drop mass':signal_jet_mass
}

signal_df = pd.DataFrame(signal_dict)
display(signal_df.head())
```

```
   # of SVs recorded  generator-level soft drop mass
0                  5                      302.250000
1                  4                       21.734375
2                  2                      155.500000
3                  0                      159.625000
4                  1                      477.500000
```

[62]:
```
_ = sns.set(rc={'figure.figsize':(12,8)})

signal_box = sns.boxplot(x='# of SVs recorded', y='generator-level soft drop␣
 ↪mass',
                        data=signal_df, palette='pastel')
```

```
_ = signal_box.grid(False)
TITLE = 'Distribution of jet mass per jet\nwith different number of recorded␣
↪SVs'
_ = signal_box.set_title(TITLE, fontdict={'size':20, 'weight':'bold', 'color':
↪'brown'})
```




As the presence of secondary vertices in a jet often indicates presence of heavy particles with longer lifespan, we expected number of secondary vertices recorded in a jet to have positive relationship with the jet mass. For instance, the two b-quarks produced from the decay of Higgs boson have relatively longer lifespan due to its heavier weight, which allow them to travel far enough from primary vertex and form secondary vertex. From above boxplots, we can see clear positive trend in jet mass for QCD jets as more secondary vertices are recorded in them. Surprisingly, signal jets failed to show as strong positive trend in jet mass with respect to increasing number of recorded secondary vertices. We strongly assume this has to do with presence of noise data in our dataset. But overall, there exist positive relationship between number of secondary vertices recorded in a jet and the mass of that jet.

[63]:
```
summary_df = pd.DataFrame({
    'Average jet mass':[avg_mass_qcd, avg_mass_signal],
    'Median jet mass':[med_qcd_jet_mass, med_signal_jet_mass]
```

```
}, index=['QCD', 'Signal'])
display(summary_df)
```

```
        Average jet mass  Median jet mass
QCD           125.160004           57.375
Signal        180.309998          109.250
```

---