

In [118]:

```
import warnings
warnings.filterwarnings("ignore")
```

## Práctica 2 Tipología y ciclo de vida de los datos

### Índice

1. Descripción del data set
2. Integración y selección de los datos de interés a analizar.
3. Limpieza de los datos y valores extremos
  - A. Limpieza de datos
  - B. Identificación y tratamiento de valores extremos
4. Análisis de los datos
  - A. Selección de los grupos de datos
  - B. Normalidad y homogeneidad de la varianza
  - C. Pruebas estadísticas
5. Representación resultados
6. Resolución del problema

### 1. Descripción del data set

El dataset elegido es un conjunto de datos formado por la información recogida a través de los experimentos que se realizarán para detectar el bosón de Higgs. Se quiere comprobar cuales son los atributos que más influyen para saber si es un bosón o no.

En lo referente a la parte técnica de los datos, el conjunto de datos tiene un número de muestras (250000) suficientes para un primer estudio, el número de atributos por muestra (30) son bastantes para el uso de los diferentes algoritmos y reglas de asociación. Por otra parte, para el estudio tenemos dos clases objetivo, **b** (bosón de Higgs) y **s** (no bosón de Higgs).

### 2. Integración y selección de los datos de interés a analizar

Antes de realizar las tareas de limpieza y acondicionamiento, se hará un análisis del dataset y los parámetros de este.

Importamos las librerías que vamos a necesitar para realizar el estudio.

In [119]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tqdm
import scipy

from sklearn import decomposition
from sklearn import linear_model
```

Primero cargamos los datos y visualizamos las 5 primeras filas para comprobar que se han cargado correctamente. Además, en la descripción del conjunto de datos se detalla que los valores -999.000 se corresponden con valores nulos.

In [120]:

```
df = pd.read_csv('training.csv', na_values=[-999.000])
df.head()
```

Out[120]:

	EventId	DER_mass_MMC	DER_mass_transverse_met_lep	DER_mass_vis	DER_pt_h	DER
0	100000	138.470		51.655	97.827	27.980
1	100001	160.937		68.768	103.235	48.146
2	100002	NaN		162.172	125.953	35.635
3	100003	143.905		81.417	80.943	0.414
4	100004	175.864		16.915	134.805	16.405

5 rows × 33 columns

Vamos a ver la dimensionalidad del dataset.

In [121]:

```
df.shape
```

Out[121]:

```
(250000, 33)
```

Vemos que el dataset contiene 8036 filas y 14 columnas.

A continuación, veremos cada uno de los parámetros del conjunto de datos.

In [122]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250000 entries, 0 to 249999
Data columns (total 33 columns):
EventId           250000 non-null int64
DER_mass_MMC      211886 non-null float64
DER_mass_transverse_met_lep 250000 non-null float64
DER_mass_vis       250000 non-null float64
DER_pt_h           250000 non-null float64
DER_deltaeta_jet_jet 72543 non-null float64
DER_mass_jet_jet   72543 non-null float64
DER_prodeta_jet_jet 72543 non-null float64
DER_deltar_tau_lep 250000 non-null float64
DER_pt_tot         250000 non-null float64
DER_sum_pt          250000 non-null float64
DER_pt_ratio_lep_tau 250000 non-null float64
DER_met_phi_centrality 250000 non-null float64
DER_lep_eta_centrality 72543 non-null float64
PRI_tau_pt          250000 non-null float64
PRI_tau_eta         250000 non-null float64
PRI_tau_phi          250000 non-null float64
PRI_lep_pt          250000 non-null float64
PRI_lep_eta         250000 non-null float64
PRI_lep_phi          250000 non-null float64
PRI_met              250000 non-null float64
PRI_met_phi          250000 non-null float64
PRI_met_sumet        250000 non-null float64
PRI_jet_num          250000 non-null int64
PRI_jet_leading_pt    150087 non-null float64
PRI_jet_leading_eta   150087 non-null float64
PRI_jet_leading_phi   150087 non-null float64
PRI_jet_subleading_pt 72543 non-null float64
PRI_jet_subleading_eta 72543 non-null float64
PRI_jet_subleading_phi 72543 non-null float64
PRI_jet_all_pt        250000 non-null float64
Weight               250000 non-null float64
Label                250000 non-null object
dtypes: float64(30), int64(2), object(1)
memory usage: 62.9+ MB
```

Podemos ver que hay dos tipos de variables **númericas float64** (30) e **int64** (2) y además un tipo **object** que se corresponde con la clase. Para todas las columnas no hay el mismo número de registros no nulos, esto oscilan entre 72543 y 250000.

Eliminamos las columnas *EventId* y *Weight* del dataset ya que contienen el identificador de cada fila y el peso asociado a la fila respectivamente. Ambas columnas no son atributos derivados del experimento.

In [123]:

```
df.drop(['EventId', 'Weight'], axis=1, inplace=True)
```

Ahora, veremos las características basicas de cada párametro.

In [124]:

```
df.describe()
```

Out[124]:

	DER_mass_MMC	DER_mass_transverse_met_lep	DER_mass_vis	DER_pt_h	DER_
count	211886.000000	250000.000000	250000.000000	250000.000000	
mean	121.858528	49.239819	81.181982	57.895962	
std	57.298157	35.344886	40.828691	63.655682	
min	9.044000	0.000000	6.329000	0.000000	
25%	91.885250	19.241000	59.388750	14.068750	
50%	112.406000	46.524000	73.752000	38.467500	
75%	135.482000	73.598000	92.259000	79.169000	
max	1192.026000	690.075000	1349.351000	2834.999000	

8 rows × 30 columns

Podemos calcular el valor medio del resto de variables para cuando *Label* valgan b y s.

In [125]:

```
data = df.query('Label == "b"').mean()  
data
```

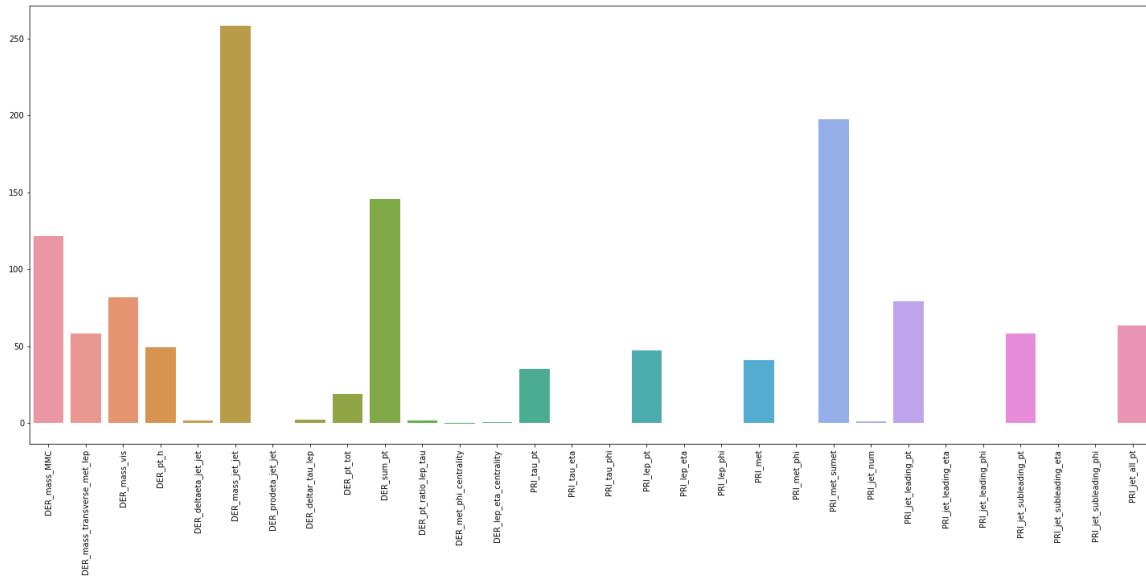
Out[125]:

DER_mass_MMC	121.325369
DER_mass_transverse_met_lep	58.208042
DER_mass_vis	81.596313
DER_pt_h	49.047438
DER_deltaeta_jet_jet	1.888940
DER_mass_jet_jet	258.415169
DER_prodeta_jet_jet	0.127851
DER_deltar_tau_lep	2.366178
DER_pt_tot	19.163180
DER_sum_pt	145.630729
DER_pt_ratio_lep_tau	1.556785
DER_met_phi_centrality	-0.362495
DER_lep_eta_centrality	0.347744
PRI_tau_pt	34.900855
PRI_tau_eta	-0.010146
PRI_tau_phi	-0.002396
PRI_lep_pt	47.169168
PRI_lep_eta	-0.020892
PRI_lep_phi	0.038132
PRI_met	41.183666
PRI_met_phi	-0.019900
PRI_met_sumet	197.419574
PRI_jet_num	0.884929
PRI_jet_leading_pt	79.391651
PRI_jet_leading_eta	-0.003724
PRI_jet_leading_phi	-0.012065
PRI_jet_subleading_pt	58.330466
PRI_jet_subleading_eta	-0.014049
PRI_jet_subleading_phi	0.008710
PRI_jet_all_pt	63.560706

dtype: float64

In [126]:

```
plt.figure(figsize=(25,10))
ax = sns.barplot(x=data.index.values, y=data.values)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()
```



En la anterior imagen se puede ver la representación en una gráfica de barras de las medias de cada columna con respecto a la columna *Label* cuando es un bosón de Higgs.

In [127]:

```
data = df.query('Label == "s"').mean()
data
```

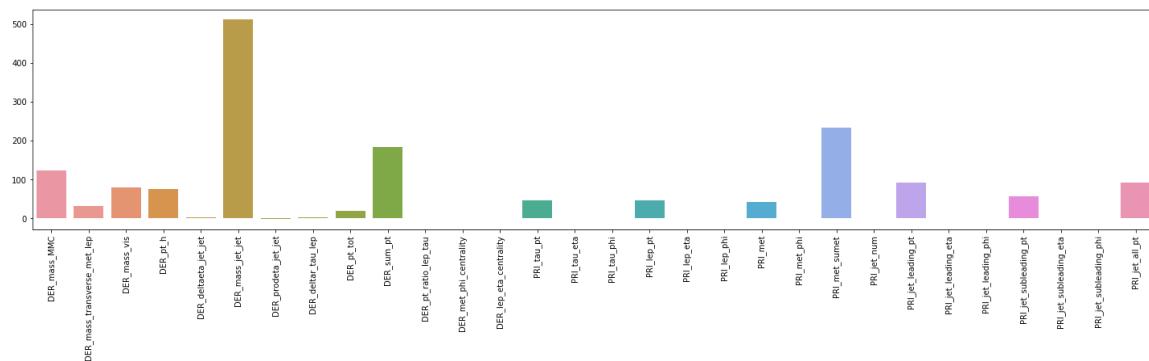
Out[127]:

DER_mass_MMC	122.689202
DER_mass_transverse_met_lep	32.036287
DER_mass_vis	80.387179
DER_pt_h	74.869878
DER_deltaeta_jet_jet	3.039249
DER_mass_jet_jet	511.736244
DER_prodeta_jet_jet	-1.993892
DER_deltar_tau_lep	2.386378
DER_pt_tot	18.445728
DER_sum_pt	182.989011
DER_pt_ratio_lep_tau	1.208997
DER_met_phi_centrality	0.320938
DER_lep_eta_centrality	0.594759
PRI_tau_pt	46.009462
PRI_tau_eta	-0.012559
PRI_tau_phi	-0.019249
PRI_lep_pt	45.683880
PRI_lep_eta	-0.016851
PRI_lep_phi	0.053923
PRI_met	42.740767
PRI_met_phi	0.008644
PRI_met_sumet	233.540845
PRI_jet_num	1.159968
PRI_jet_leading_pt	92.936154
PRI_jet_leading_eta	-0.002604
PRI_jet_leading_phi	-0.012882
PRI_jet_subleading_pt	56.875826
PRI_jet_subleading_eta	-0.009124
PRI_jet_subleading_phi	-0.014288
PRI_jet_all_pt	91.295672

dtype: float64

In [128]:

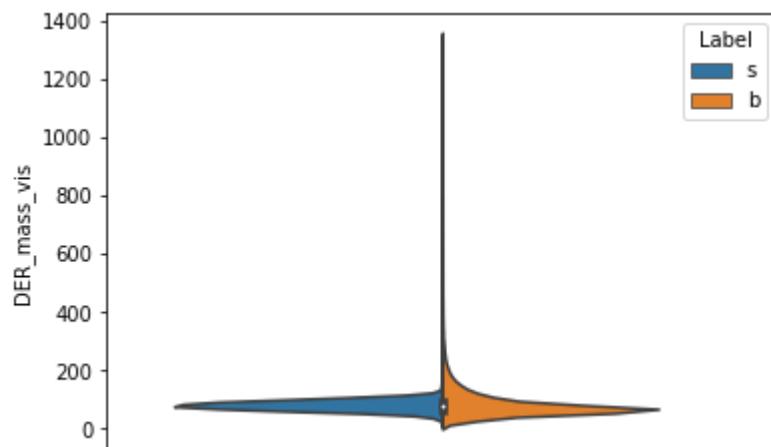
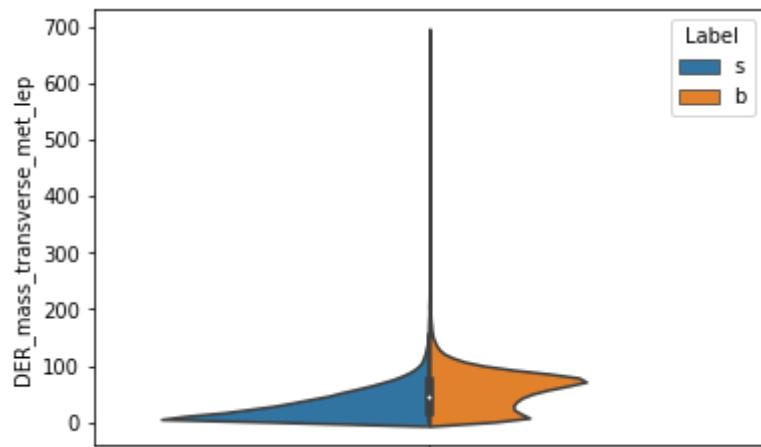
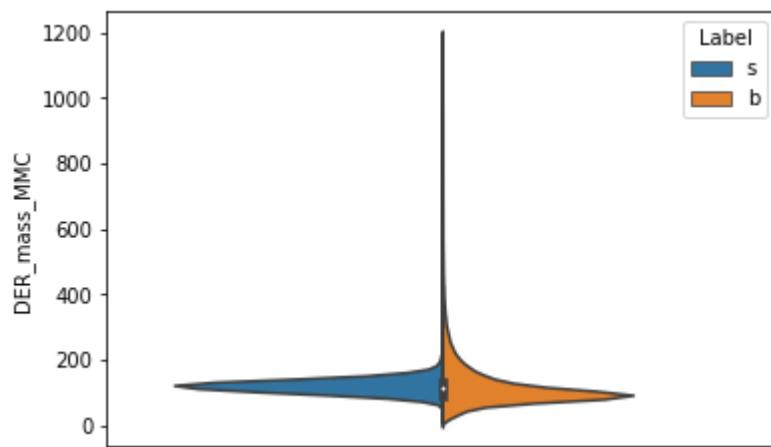
```
plt.figure(figsize=(25,5))
ax = sns.barplot(x=data.index.values, y=data.values)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
plt.show()
```

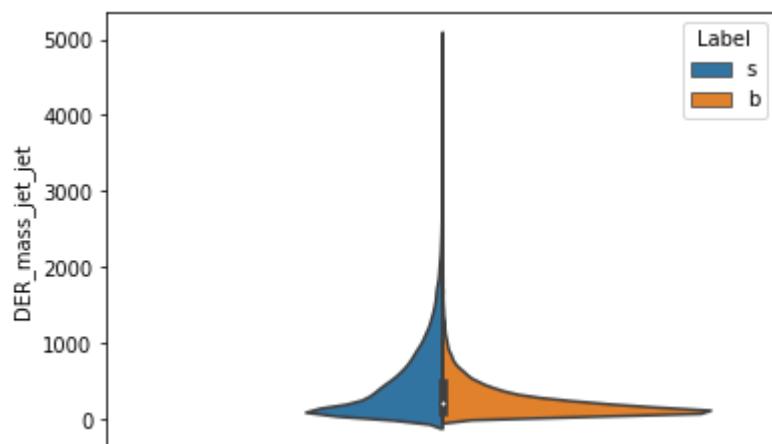
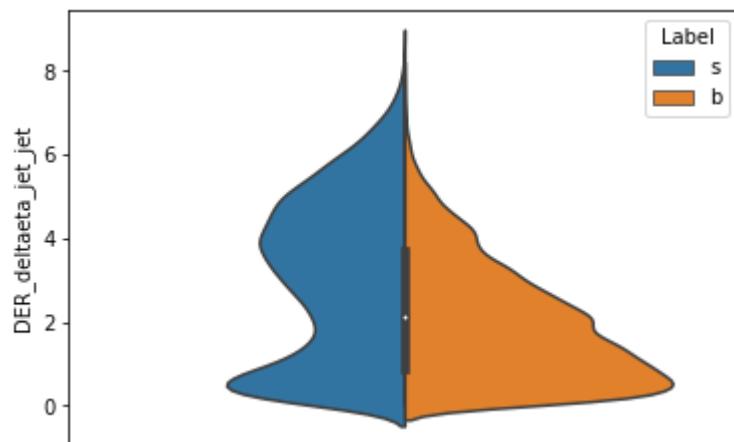
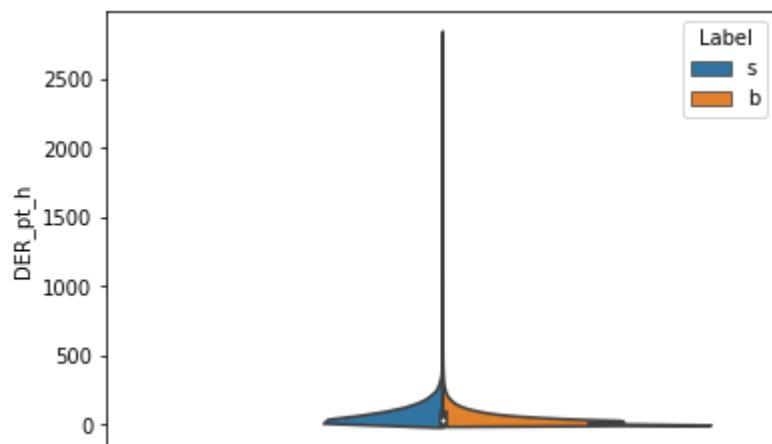


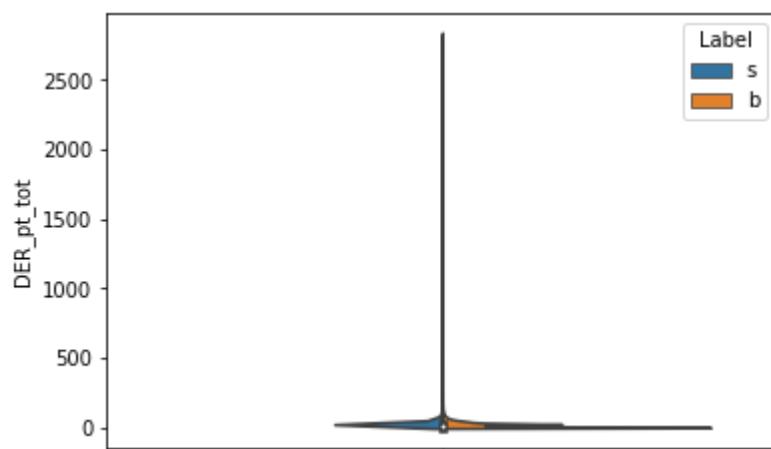
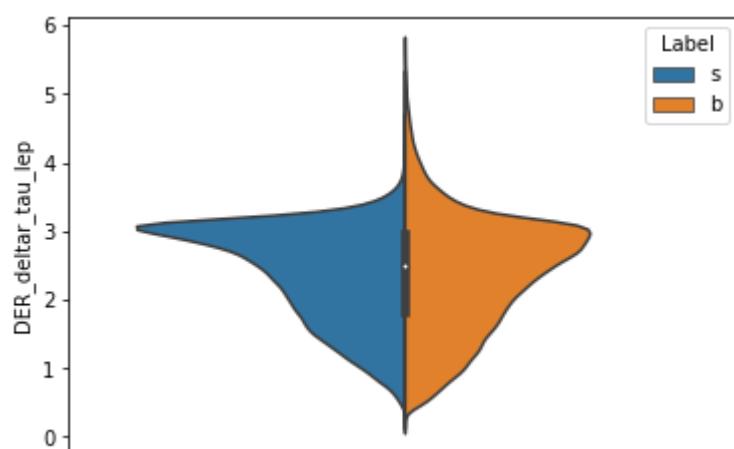
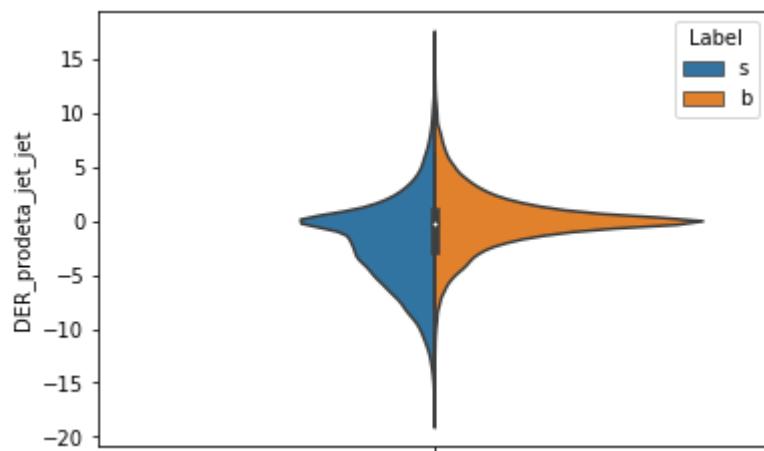
En la anterior imagen se puede ver la representación en una gráfica de barras de las medias de cada columna con respecto a la columna *Label* cuando no es un bosón de Higgs.

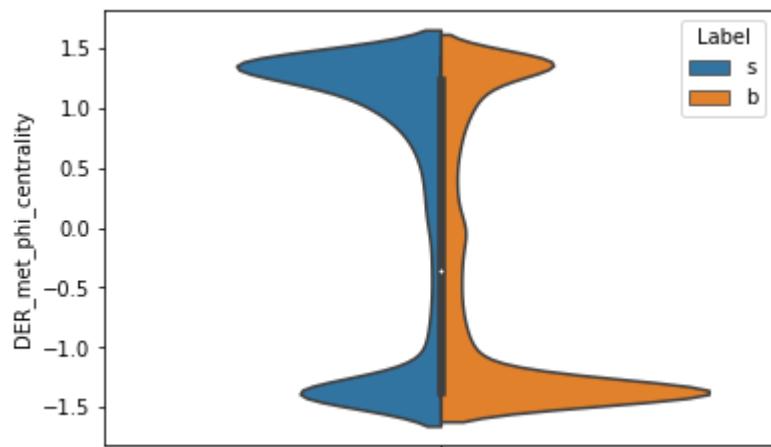
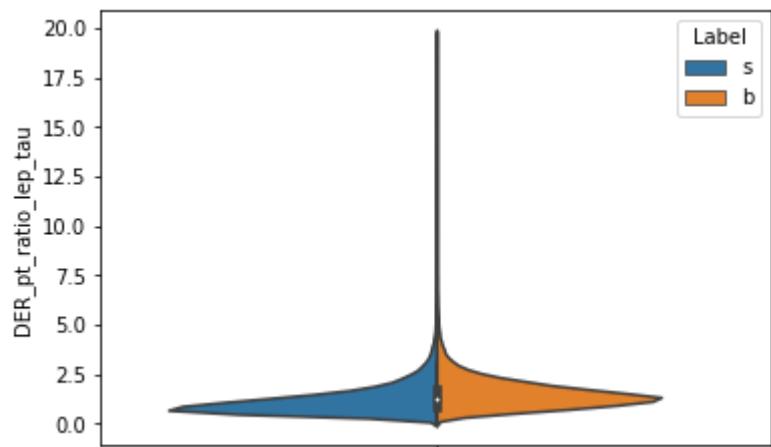
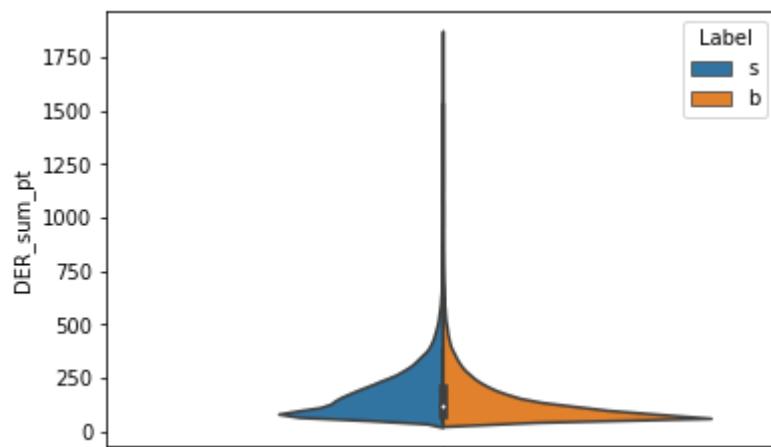
In [129]:

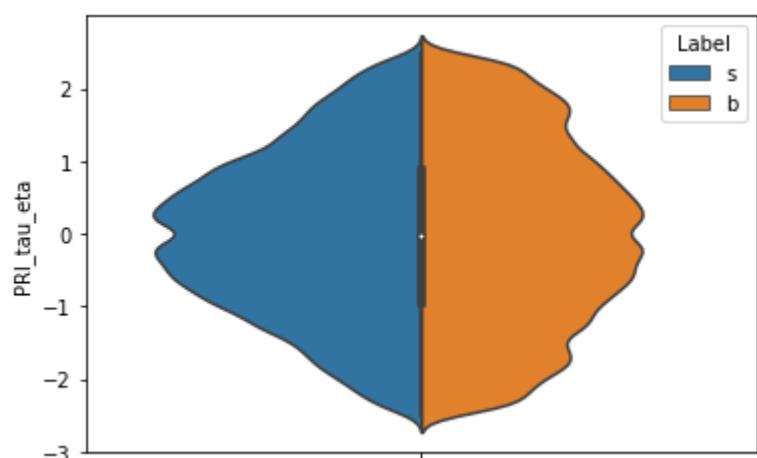
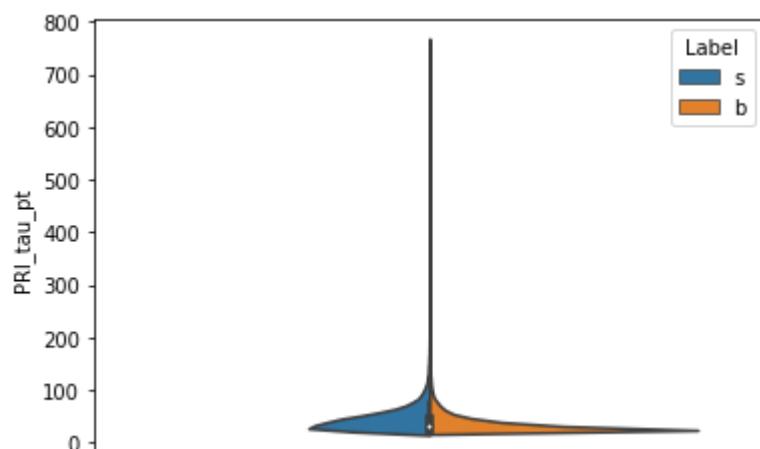
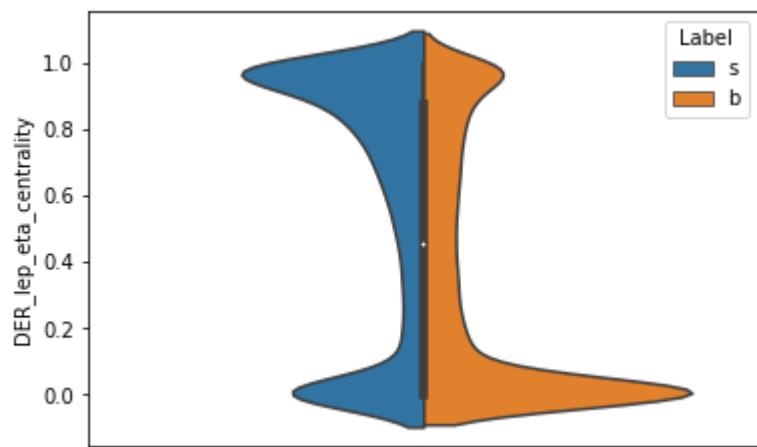
```
df['all'] = ''  
for column in df.columns:  
    if column not in ['Label', 'all']:  
        ax = sns.violinplot(x='all', y=column, hue='Label', data=df, split=True)  
        ax.set_xlabel('')  
        plt.show()  
del(df['all'])
```

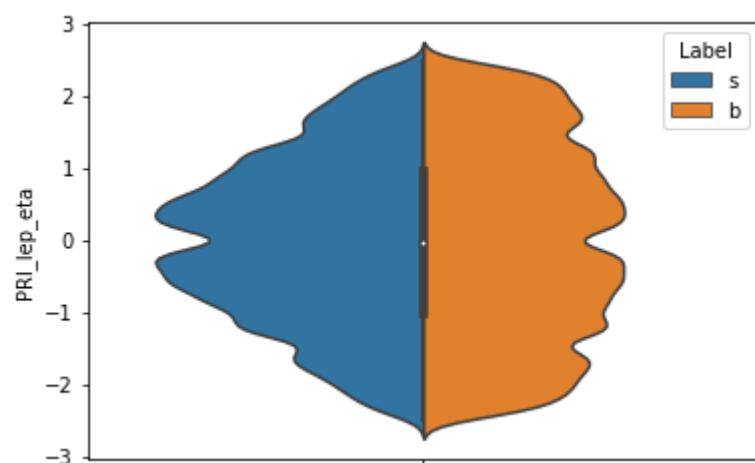
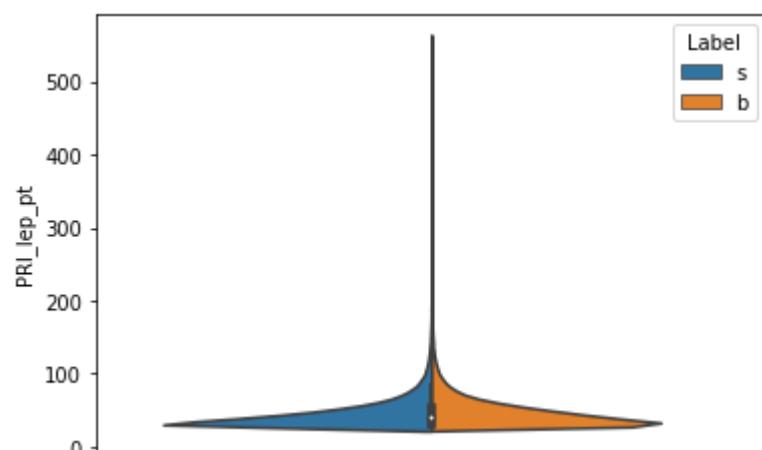
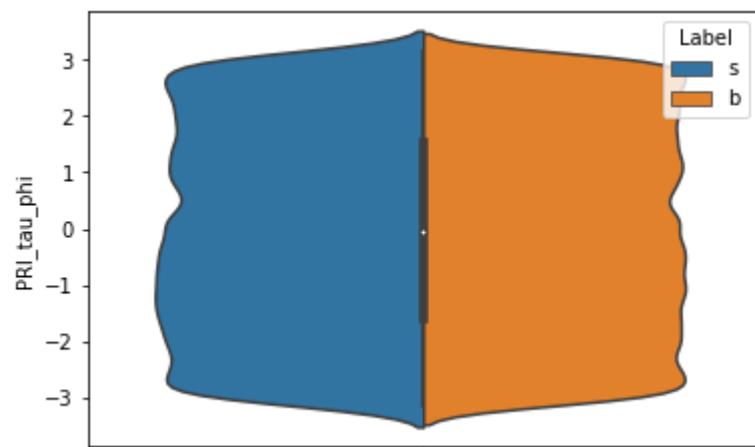


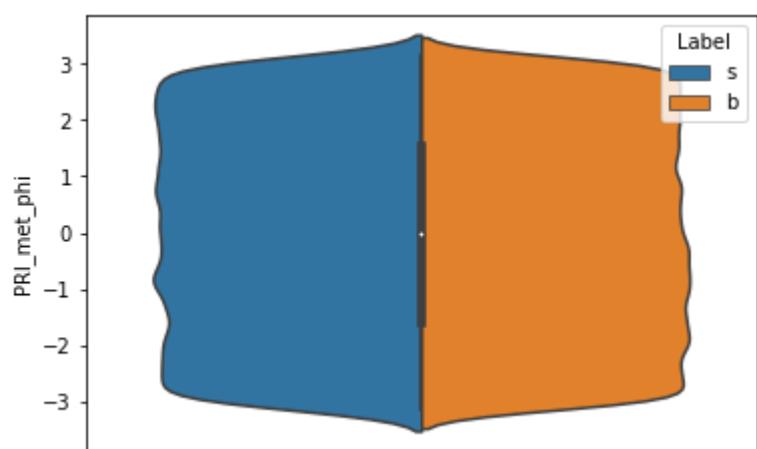
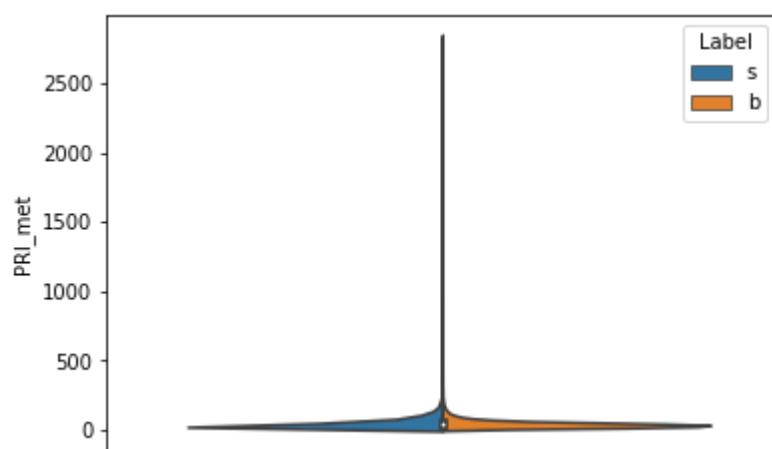
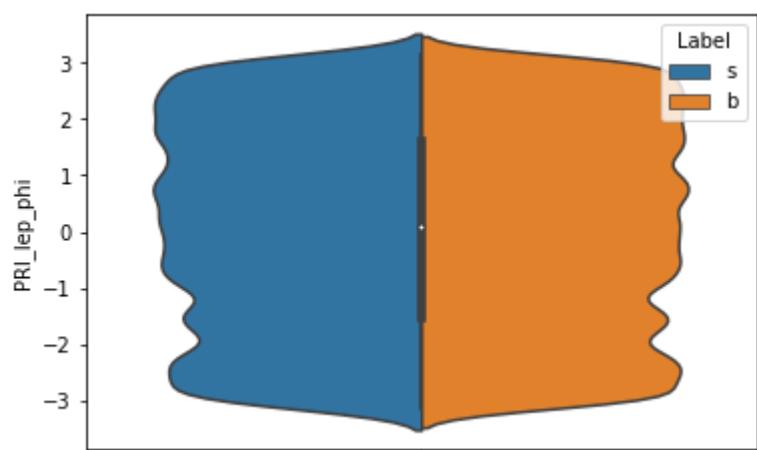


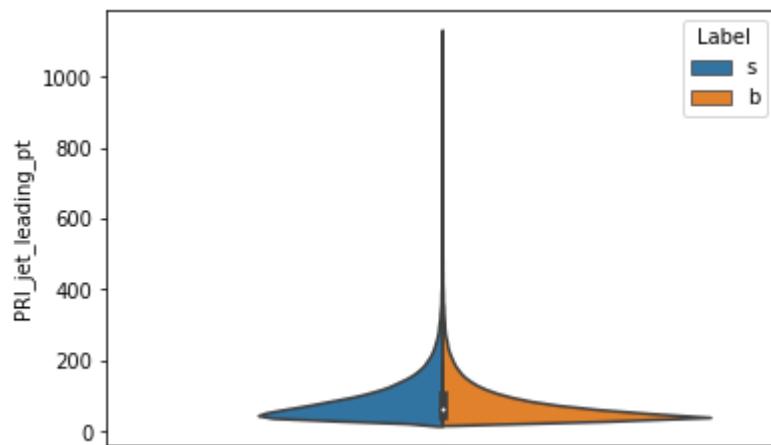
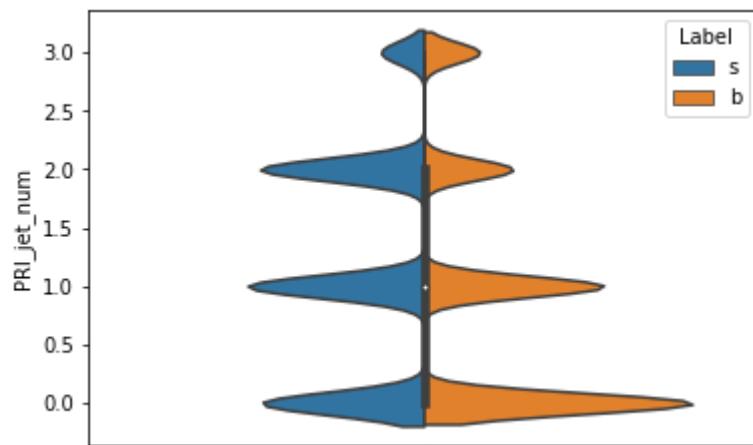
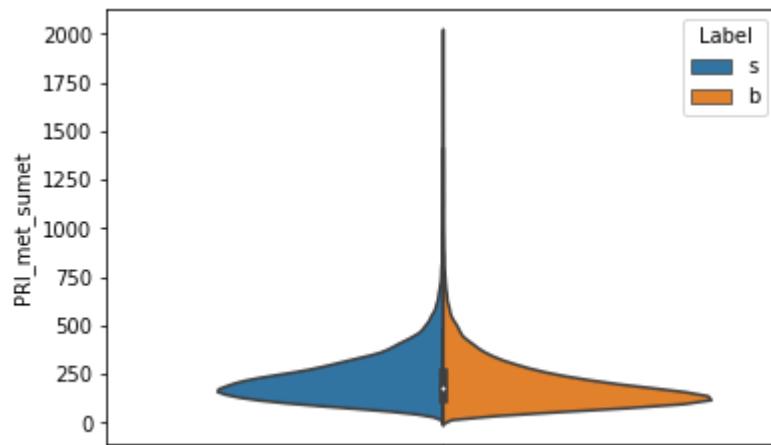


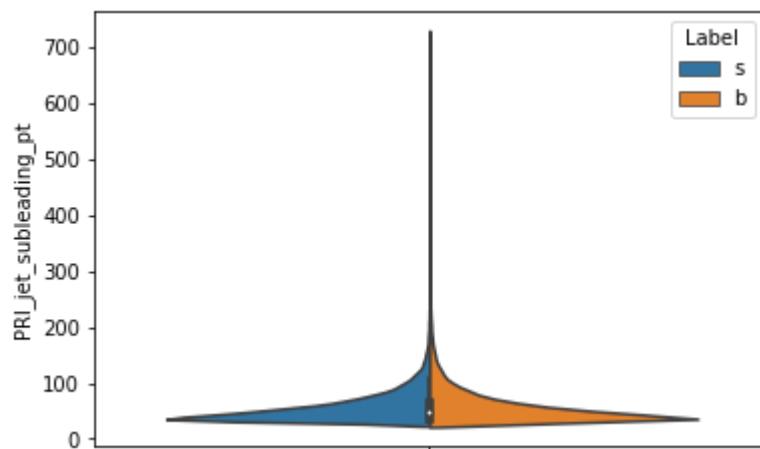
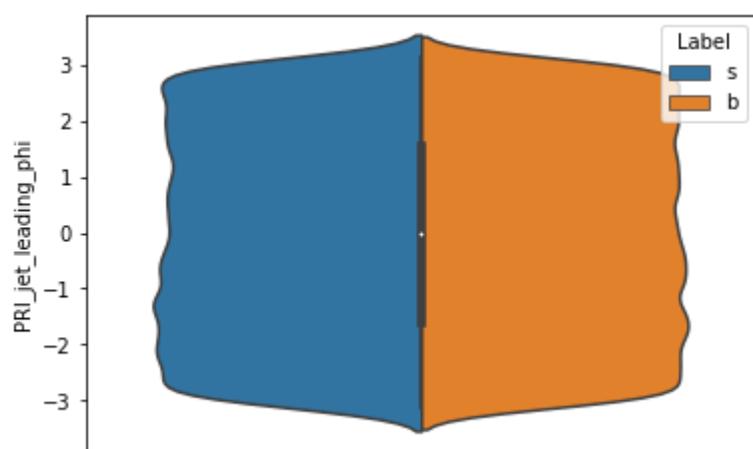
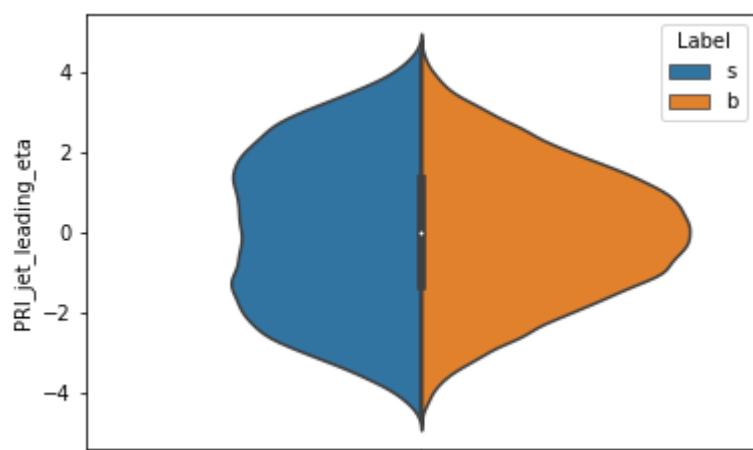


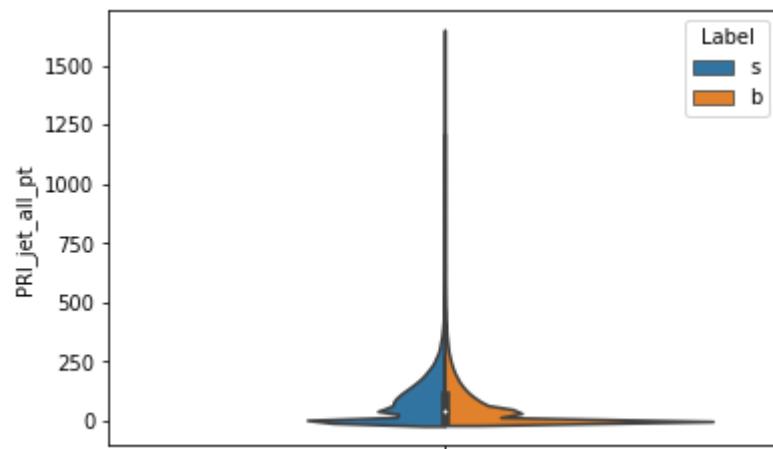
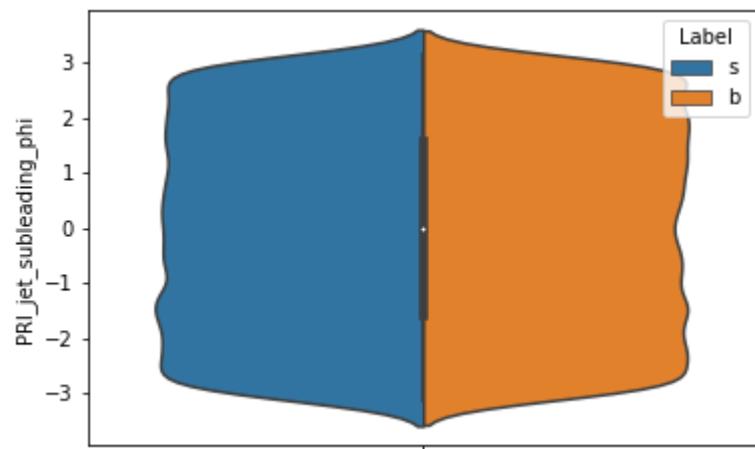
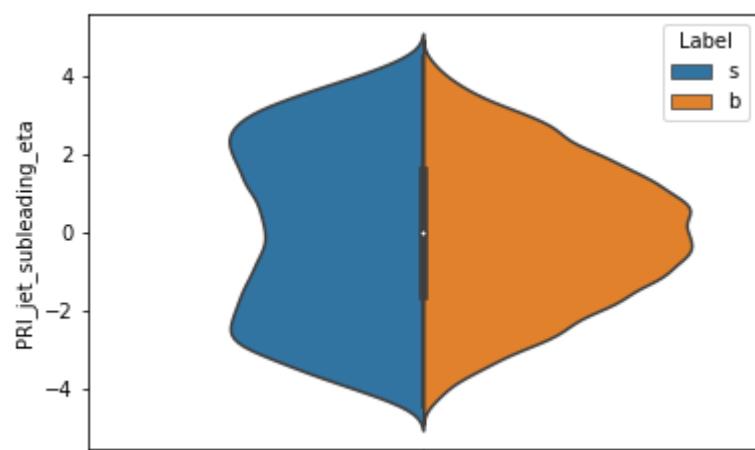








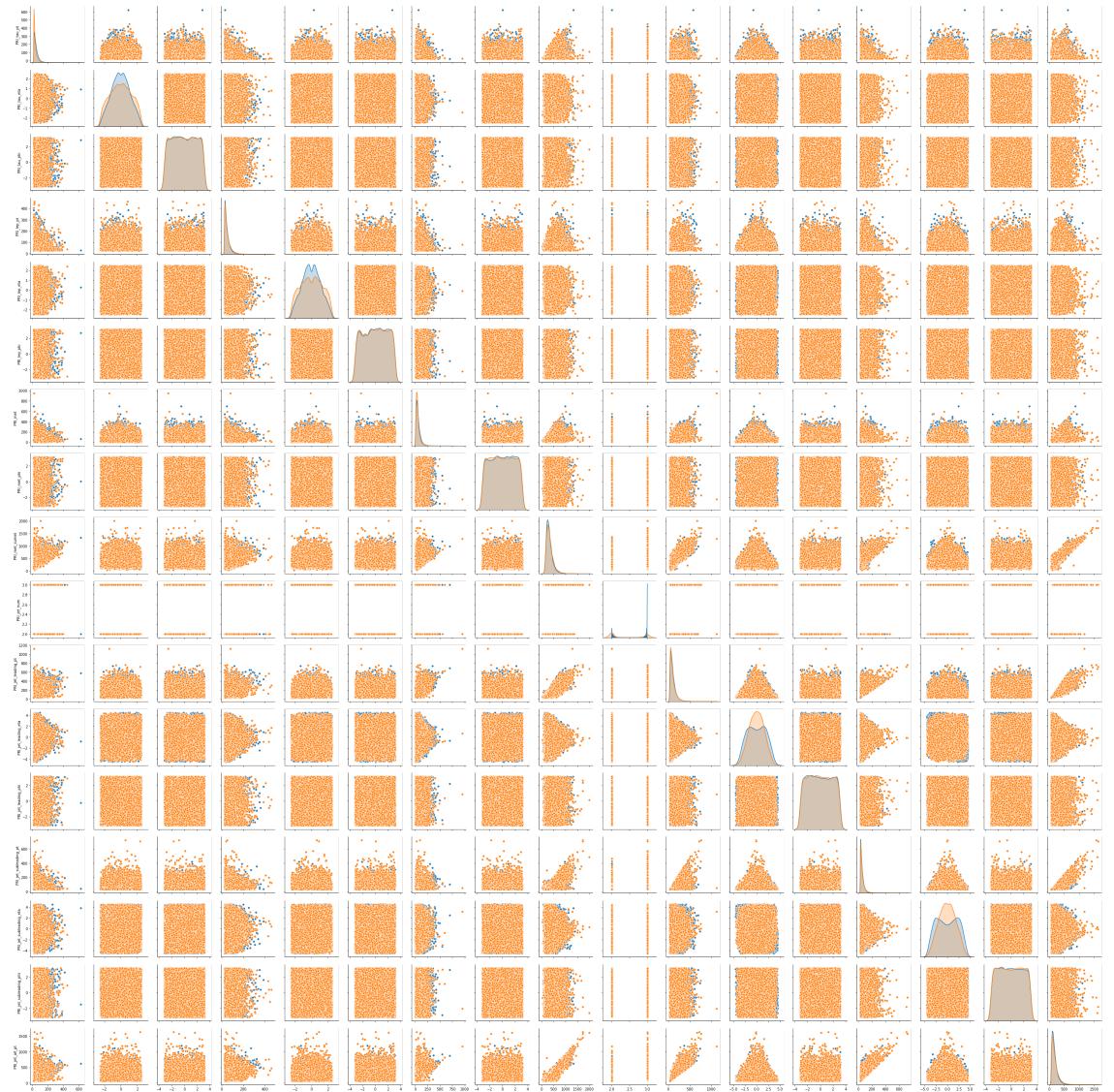




En las imágenes superiores se muestran *violin plots* para cada columna y cada clase. Estas gráficas muestran los histogramas enfrentados para la clase **b** (bosón de Higgs) y la clase **s** (no bosón de Higgs) de cada columna. Podemos ver que hay algunas columnas en las que las distribuciones son claramente diferentes como **DER\_deltaeta\_jet\_jet**, y otras en las que son prácticamente iguales como **PRI\_jet\_subleading\_phi**. Además, algunas variables tienen una distribución uniforme (**DER\_deltaeta\_jet\_jet**) por lo que deberían ser excluidas de la parte analítica al no aportar información. También, las gráficas nos muestran tanto la media (punto blanco) como el rango intercuartílico (barra negra) de cada columna.

In [130]:

```
g = sns.pairplot(df.dropna(), vars=[x for x in df.columns if 'PRI_' in x], hue='Label')
```

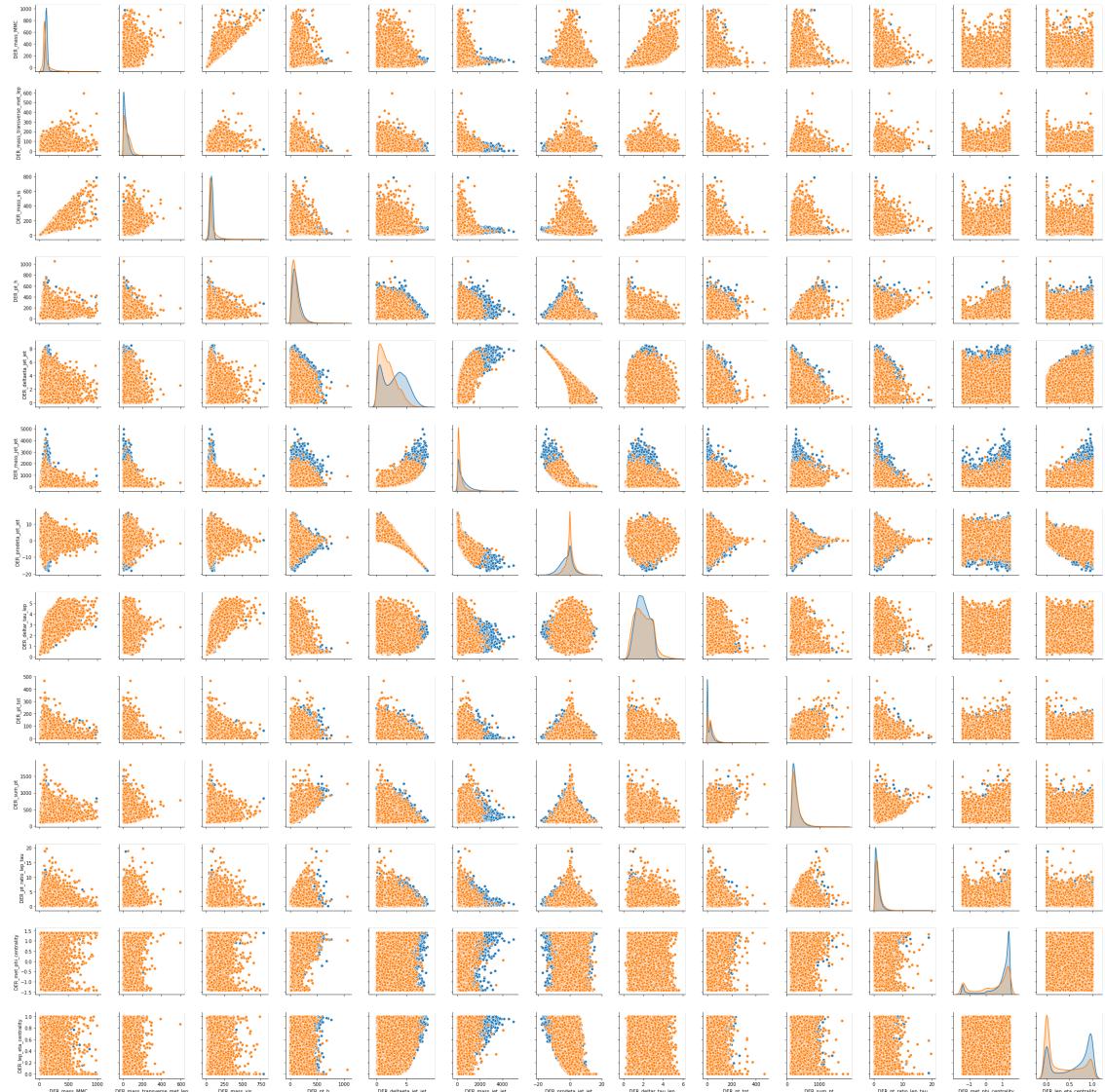


En la imagen superior se muestra un *pair plot* de las variables *raw* (las columnas identificadas con *PRI*). Esta gráfica muestra por un lado los *scatter plot* de todas las columnas contra todas las columnas y en la diagonal la distribución de la columna en cuestión. Además, en color naranja se tienen los valores para la clase **b** y en azul para la clase **s**. En esta gráfica se puede observar lo siguiente:

- Las variables cuya distribución es uniforme se identifican mas facilmente que en el *violin plot* (*PRI\_jet\_subleading\_phi* p.e.)
- Se puede ver que hay cierta correlación entre variables como *PRI\_met\_sumet* y *PRI\_get\_all\_pt* y otras en las que no hay correlación como *PRI\_jet\_subleading\_phi* y *PRI\_tau\_eta*.
- Hay variables binarias como *PRI\_jet\_num*.
- En ningún *scatter plot* se puede hacer una diferenciación clara de la clase.

In [131]:

```
g = sns.pairplot(df.dropna(), vars=[x for x in df.columns if 'DER_' in x], hue='Label')
```



En la imagen superior se muestra un *pair plot* de las variables derivadas de las *raw* (las columnas identificadas con *DER*). En esta gráfica se puede observar lo siguiente:

- No hay ninguna columna cuya distribución sea uniforme.
- Se puede ver que hay cierta correlación entre variables como *DER\_mass\_vis* y *DER\_mass\_MMC* y otras en las que no hay correlación como *DER\_lep\_eta\_centrality* y *DER\_met\_phi\_centrality*.
- No hay variables binarias.
- Se puede hacer una diferenciación clara de la clase en algunos *scatter plot* como *DER\_proleta\_jet\_jet* y *DER\_mass\_jet\_jet*.

## 3. Limpieza de los datos y valores extremos

Limparemos los parámetros que no sean útiles para el estudio, reacondicionaremos los que sean necesarios y exploraremos los valores extremos.

### A. Limpieza de datos

Se eliminan aquellos parámetros que no son válidos o no aportan información al estudio identificados en el punto anterior. A estos parámetros hay que sumarle *EventId* y *Weight* que ya fueron eliminados anteriormente.

In [132]:

```
to_drop = [
    'PRI_tau_phi',
    'PRI_lep_phi',
    'PRI_met_phi',
    'PRI_jet_leading_phi',
    'PRI_jet_subleading_phi',
]
df.drop(to_drop, axis=1, inplace=True)
```

Transformamos la columna *Label* a binario.

In [133]:

```
df['Label'] = df.Label.apply(lambda x: 1 if x is 'b' else 0)
```

El conjunto de datos contiene numerosos valores a nulo, que necesitan o bien ser reemplazados por un valor o bien eliminados. En este caso dado el gran número de valores nulos por cada columna se ha decidido sustituirlos. De entre las técnicas recomendadas para realizar la sustitución, la mejor sin duda es utilizar la regresión lineal. En este caso, por cada columna que presenta valores nulos, se ha entrenado un regresor lineal a partir de los datos no nulos para obtener los valores con los que sustituirlos.

In [134]:

```
with_na = df.columns[df.isna().any()].values
without_na = df.columns[~df.isna().any()].values
```

In [135]:

```
for na_column in tqdm.tqdm(with_na):
    X = df.dropna(subset=[na_column])[without_na].values
    y = df.dropna(subset=[na_column])[na_column].values
    model = linear_model.LinearRegression().fit(X, y)
    df[na_column] = df.apply(lambda row: model.predict([row[without_na]])[0] if np.isnan(row[na_column]) else row[na_column], axis=1)
```

100%|██████████| 9/9 [12:15&lt;00:00, 81.77s/it]

In [136]:

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250000 entries, 0 to 249999
Data columns (total 26 columns):
DER_mass_MMC           250000 non-null float64
DER_mass_transverse_met_lep 250000 non-null float64
DER_mass_vis            250000 non-null float64
DER_pt_h                250000 non-null float64
DER_deltaeta_jet_jet   250000 non-null float64
DER_mass_jet_jet        250000 non-null float64
DER_prodeta_jet_jet    250000 non-null float64
DER_deltar_tau_lep     250000 non-null float64
DER_pt_tot               250000 non-null float64
DER_sum_pt               250000 non-null float64
DER_pt_ratio_lep_tau   250000 non-null float64
DER_met_phi_centrality 250000 non-null float64
DER_lep_eta_centrality 250000 non-null float64
PRI_tau_pt              250000 non-null float64
PRI_tau_eta              250000 non-null float64
PRI_lep_pt              250000 non-null float64
PRI_lep_eta              250000 non-null float64
PRI_met                  250000 non-null float64
PRI_met_sumet            250000 non-null float64
PRI_jet_num              250000 non-null int64
PRI_jet_leading_pt       250000 non-null float64
PRI_jet_leading_eta      250000 non-null float64
PRI_jet_subleading_pt    250000 non-null float64
PRI_jet_subleading_eta   250000 non-null float64
PRI_jet_all_pt            250000 non-null float64
Label                   250000 non-null int64
dtypes: float64(24), int64(2)
memory usage: 49.6 MB
```

Como se puede ver ya no hay valores nulos en nuestro conjunto de datos.

El siguiente paso es detectar y eliminar *outliers* en nuestro dataset.

## B. Identificación y tratamiento de valores extremos

In [137]:

```
columns = [x for x in df.columns if x != 'Label']
Q1 = df[columns].quantile(0.25)
Q3 = df[columns].quantile(0.75)
IQR = Q3 - Q1
df_final = df[~((df[columns] < (Q1 - 1.5 * IQR)) | (df[columns] > (Q3 + 1.5 * IQR))).any(axis=1)]
print(df_final.shape)
```

(147421, 26)

El método elegido para detectar y eliminar los *outliers* es el metodo intercuartílico. Aplicar este método nos deja un total de 147421 filas (se han eliminado mas de 100000 filas con *outliers*).

In [204]:

```
df_final.to_csv(path_or_buf='./df_final.csv', index=False)
```

## 4. Análisis de los datos

### A. Selección de los grupos de datos

Todas la variables de nuestro conjunto de datos son cuantitativas y todas pueden influir en saber si es un boson de Higgs o no, por eso tomaremos todas las columnas restantes como un solo grupo.

### B. Normalidad y homogeneidad de la varianza

Primero comprobaremos si la distribución de las variables es normal.

In [165]:

```
for col in df_final.columns:  
    print(col)  
    print(scipy.stats.anderson(df_final[col], dist='norm'))
```

```
DER_mass_MMC
AndersonResult(statistic=98.5481432141969, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_mass_transverse_met_lep
AndersonResult(statistic=1251.787850568886, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_mass_vis
AndersonResult(statistic=275.1965640685812, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_pt_h
AndersonResult(statistic=4894.509780603519, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_deltaeta_jet_jet
AndersonResult(statistic=4815.993620498542, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_mass_jet_jet
AndersonResult(statistic=5268.474837162474, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_prodeta_jet_jet
AndersonResult(statistic=9659.18560780058, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_deltar_tau_lep
AndersonResult(statistic=2290.6781340038287, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_pt_tot
AndersonResult(statistic=7335.18541702244, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_sum_pt
AndersonResult(statistic=4519.32373488904, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_pt_ratio_lep_tau
AndersonResult(statistic=1132.408189968497, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_met_phi_centrality
AndersonResult(statistic=17047.25690866416, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
DER_lep_eta_centrality
AndersonResult(statistic=8489.307141200843, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_tau_pt
AndersonResult(statistic=5211.504185609228, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_tau_eta
AndersonResult(statistic=417.00659211110906, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_lep_pt
```

```

AndersonResult(statistic=3124.397287285945, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_lep_eta
AndersonResult(statistic=726.9293574551994, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_met
AndersonResult(statistic=913.7573028988263, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_met_sumet
AndersonResult(statistic=966.7922132867388, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_jet_num
AndersonResult(statistic=21789.273493743676, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_jet_leading_pt
AndersonResult(statistic=19652.3351637795, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_jet_leading_eta
AndersonResult(statistic=1189.3133968000766, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_jet_subleading_pt
AndersonResult(statistic=7196.762960640393, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_jet_subleading_eta
AndersonResult(statistic=329.84317528278916, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
PRI_jet_all_pt
AndersonResult(statistic=17202.235595131642, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))
Label
AndersonResult(statistic=32404.888301906438, critical_values=array([0.576,
0.656, 0.787, 0.918, 1.092]), significance_level=array([15. , 10. , 5. ,
2.5, 1. ]))

```

Como se puede ver en los resultados de la prueba de normalidad de Anderson-Darling, en todas las variables los statistic test son mayores que los critical values por lo tanto no siguen una distribución normal.

Ahora comprobaremos la homogeneidad de la varianza con el test de Fligner-Killeen.

In [166]:

```
for col in df_final.columns:
    if col != 'Label':
        print(col)
        print(scipy.stats.fligner(df_final['Label'], df[col]))
```

DER\_mass\_MMC  
FlignerResult(statistic=18182.64433489327, pvalue=0.0)  
DER\_mass\_transverse\_met\_lep  
FlignerResult(statistic=62619.96998300417, pvalue=0.0)  
DER\_mass\_vis  
FlignerResult(statistic=17013.9791803394, pvalue=0.0)  
DER\_pt\_h  
FlignerResult(statistic=96846.33869801769, pvalue=0.0)  
DER\_deltaeta\_jet\_jet  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
DER\_mass\_jet\_jet  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
DER\_prodeta\_jet\_jet  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
DER\_deltar\_tau\_lep  
FlignerResult(statistic=17500.93108279067, pvalue=0.0)  
DER\_pt\_tot  
FlignerResult(statistic=115873.66705675346, pvalue=0.0)  
DER\_sum\_pt  
FlignerResult(statistic=26658.525038467647, pvalue=0.0)  
DER\_pt\_ratio\_lep\_tau  
FlignerResult(statistic=24094.271884370457, pvalue=0.0)  
DER\_met\_phi\_centrality  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
DER\_lep\_eta\_centrality  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
PRI\_tau\_pt  
FlignerResult(statistic=16171.177026792644, pvalue=0.0)  
PRI\_tau\_eta  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
PRI\_lep\_pt  
FlignerResult(statistic=12578.813458124798, pvalue=0.0)  
PRI\_lep\_eta  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
PRI\_met  
FlignerResult(statistic=45214.979657922224, pvalue=0.0)  
PRI\_met\_sumet  
FlignerResult(statistic=26975.623748596012, pvalue=0.0)  
PRI\_jet\_num  
FlignerResult(statistic=77806.0381201853, pvalue=0.0)  
PRI\_jet\_leading\_pt  
FlignerResult(statistic=31925.191343641625, pvalue=0.0)  
PRI\_jet\_leading\_eta  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
PRI\_jet\_subleading\_pt  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
PRI\_jet\_subleading\_eta  
FlignerResult(statistic=204598.00452110157, pvalue=0.0)  
PRI\_jet\_all\_pt  
FlignerResult(statistic=165673.53999610778, pvalue=0.0)

Puesto que todos los p-valor son 0 (menores que 0.05) se acepta que ninguna columna es homogenea con la columna Label.

## C. Pruebas estadísticas

En primer lugar haremos un análisis de correlación entre las distintas variables para ver cual influye más en la variable Label, ya que hemos comprobado que estas no siguen una distribución normal.

In [167]:

```
for col in df_final.columns:  
    if col != 'Label':  
        print(col)  
        print(scipy.stats.spearmanr(df_final['Label'], df_final[col]))
```

```
DER_mass_MMC
SpearmanrResult(correlation=-0.29218521887422694, pvalue=0.0)
DER_mass_transverse_met_lep
SpearmanrResult(correlation=0.415623597374044, pvalue=0.0)
DER_mass_vis
SpearmanrResult(correlation=-0.25673318331054545, pvalue=0.0)
DER_pt_h
SpearmanrResult(correlation=-0.09242742652245078, pvalue=5.091463866462e-2
77)
DER_deltaeta_jet_jet
SpearmanrResult(correlation=-0.7361986888714129, pvalue=0.0)
DER_mass_jet_jet
SpearmanrResult(correlation=-0.7202372035135184, pvalue=0.0)
DER_prodeta_jet_jet
SpearmanrResult(correlation=0.7478176360611252, pvalue=0.0)
DER_deltar_tau_lep
SpearmanrResult(correlation=-0.23827247938606516, pvalue=0.0)
DER_pt_tot
SpearmanrResult(correlation=-0.03432294554723078, pvalue=1.110753229306439
6e-39)
DER_sum_pt
SpearmanrResult(correlation=-0.19304801590941648, pvalue=0.0)
DER_pt_ratio_lep_tau
SpearmanrResult(correlation=0.32022598070406366, pvalue=0.0)
DER_met_phi_centrality
SpearmanrResult(correlation=-0.13373680170020455, pvalue=0.0)
DER_lep_eta_centrality
SpearmanrResult(correlation=-0.7463482413405917, pvalue=0.0)
PRI_tau_pt
SpearmanrResult(correlation=-0.36373403384967773, pvalue=0.0)
PRI_tau_eta
SpearmanrResult(correlation=-0.003997203781955716, pvalue=0.12484839564376
35)
PRI_lep_pt
SpearmanrResult(correlation=0.06354176763220992, pvalue=1.0080867923719763
e-131)
PRI_lep_eta
SpearmanrResult(correlation=-0.00672914751570917, pvalue=0.009774867408845
351)
PRI_met
SpearmanrResult(correlation=0.22453368129058307, pvalue=0.0)
PRI_met_sumet
SpearmanrResult(correlation=-0.14919012496506415, pvalue=0.0)
PRI_jet_num
SpearmanrResult(correlation=-0.0778592175118245, pvalue=6.011019369079618e
-197)
PRI_jet_leading_pt
SpearmanrResult(correlation=-0.09994953811485402, pvalue=0.0)
PRI_jet_leading_eta
SpearmanrResult(correlation=-0.004122976939491147, pvalue=0.11341495903201
877)
PRI_jet_subleading_pt
SpearmanrResult(correlation=-0.33735058719687167, pvalue=0.0)
PRI_jet_subleading_eta
SpearmanrResult(correlation=-0.02065318440532039, pvalue=2.179840681215935
4e-15)
PRI_jet_all_pt
SpearmanrResult(correlation=-0.09868655908423346, pvalue=0.0)
```

Como se puede observar las variables que más correlación tienen con Label son DER\_deltaeta\_jet\_jet (-0.5386173764979256), DER\_mass\_jet\_jet (-0.5262585853846985), DER\_proleta\_jet\_jet (0.5413863257147407) y DER\_lep\_eta\_centrality (-0.5463095548254712).

Planteamos ahora la hipótesis nula sobre todas las variables de que es más probable que sea un boson de Higgs cuanto mayor sea el valor de esta, para ello utilizamos el test no paramétrico de Mann-Whitney.

In [168]:

```
boson = df_final[df_final['Label'] == 1]
no_boson = df_final[df_final['Label'] == 0]
for col in df_final.columns:
    if col != 'Label':
        print(col)
        print(scipy.stats.mannwhitneyu(boson[col], no_boson[col], alternative='less'))
```

```
DER_mass_MMC
MannwhitneyuResult(statistic=1427615162.5, pvalue=0.0)
DER_mass_transverse_met_lep
MannwhitneyuResult(statistic=3454725287.5, pvalue=1.0)
DER_mass_vis
MannwhitneyuResult(statistic=1529147071.5, pvalue=0.0)
DER_pt_h
MannwhitneyuResult(statistic=1999706148.5, pvalue=3.790999538343598e-276)
DER_deltaeta_jet_jet
MannwhitneyuResult(statistic=155994662.5, pvalue=0.0)
DER_mass_jet_jet
MannwhitneyuResult(statistic=201707131.5, pvalue=0.0)
DER_prodeta_jet_jet
MannwhitneyuResult(statistic=4406103518.5, pvalue=1.0)
DER_deltar_tau_lep
MannwhitneyuResult(statistic=1582017221.0, pvalue=0.0)
DER_pt_tot
MannwhitneyuResult(statistic=2166112922.5, pvalue=5.8419717708419e-40)
DER_sum_pt
MannwhitneyuResult(statistic=1711536503.0, pvalue=0.0)
DER_pt_ratio_lep_tau
MannwhitneyuResult(statistic=3181513685.5, pvalue=1.0)
DER_met_phi_centrality
MannwhitneyuResult(statistic=1881452817.0, pvalue=0.0)
DER_lep_eta_centrality
MannwhitneyuResult(statistic=126927124.0, pvalue=0.0)
PRI_tau_pt
MannwhitneyuResult(statistic=1222704849.0, pvalue=0.0)
PRI_tau_eta
MannwhitneyuResult(statistic=2252963519.5, pvalue=0.06242399993475417)
PRI_lep_pt
MannwhitneyuResult(statistic=2446389949.5, pvalue=1.0)
PRI_lep_eta
MannwhitneyuResult(statistic=2245139442.5, pvalue=0.004887662845572987)
PRI_met
MannwhitneyuResult(statistic=2907458434.0, pvalue=1.0)
PRI_met_sumet
MannwhitneyuResult(statistic=1837142136.5, pvalue=0.0)
PRI_jet_num
MannwhitneyuResult(statistic=2074576621.5, pvalue=1.167286509432342e-196)
PRI_jet_leading_pt
MannwhitneyuResult(statistic=1978163398.0, pvalue=0.0)
PRI_jet_leading_eta
MannwhitneyuResult(statistic=2252603314.5, pvalue=0.056707336024389654)
PRI_jet_subleading_pt
MannwhitneyuResult(statistic=1298265014.0, pvalue=0.0)
PRI_jet_subleading_eta
MannwhitneyuResult(statistic=2205262069.0, pvalue=1.0970133550125944e-15)
PRI_jet_all_pt
MannwhitneyuResult(statistic=2016992196.0, pvalue=0.0)
```

Vemos que para DER\_mass\_MMC, DER\_mass\_vis, DER\_pt\_h, DER\_deltaeta\_jet\_jet, DER\_mass\_jet\_jet, DER\_deltar\_tau\_lep, DER\_pt\_tot, DER\_sum\_pt, DER\_met\_phi\_centrality, DER\_lep\_eta\_centrality, PRI\_tau\_pt, PRI\_lep\_eta, PRI\_met\_sumet, PRI\_jet\_num, PRI\_jet\_leading\_pt, PRI\_jet\_subleading\_pt y PRI\_jet\_all\_pt están por debajo del valor de significación 0.05 por lo que podemos decir que la variable Label influye sobre ellas y podemos utilizarlas para saber si es o no un bosón de Higgs.

Por último crearemos un modelo de regresión lineal para poder predecir con los valores obtenidos si es un bosón de Higgs o no.

In [192]:

```
X = df_final[:-100000].drop(df.columns.difference(['DER_deltaeta_jet_jet', 'DER_mass_jet_jet', 'DER_prodelta_jet_jet', 'DER_lep_eta_centrality']), axis=1).values
y = df_final[:-100000].drop(df.columns.difference(['Label']), axis=1).values
model1 = linear_model.LinearRegression().fit(X, y)
model1.coef_
```

Out[192]:

```
array([[ 0.66004439, -0.00094562,  0.60706121, -0.68803435]])
```

In [193]:

```
X = df_final[:-100000].drop(['Label'], axis=1).values
y = df_final[:-100000].drop(df.columns.difference(['Label']), axis=1).values
model2 = linear_model.LinearRegression().fit(X, y)
model2.coef_
```

Out[193]:

```
array([[-1.21053667e-04, -8.46092068e-05,  1.28371311e-03,
       -3.02257748e-03,  5.92935852e-01, -3.61433163e-03,
       2.95950577e-01, -1.30738713e-01, -3.17106739e-03,
       3.77321542e+00,  1.39306265e-02,  5.64128151e-02,
      -5.88183644e-01, -3.77295219e+00, -2.69493856e-03,
      -3.77379312e+00,  2.90553652e-03,  9.99280403e-05,
      -1.42760958e-03, -6.11454463e-01, -1.99260595e-03,
       1.73729808e-03, -6.80592045e-03,  2.96937954e-03,
      -3.76209933e+00]])
```

En el primer modelo solo utilizamos las variables con mejor correlación y en el segundo modelo se utilizan todas las variables, se puede comprobar por los coeficientes de determinación que el modelo con todas las variables tiene mejores coeficientes para las variables con más correlación que el otro. A continuación se muestra una predicción con el segundo modelo.

In [194]:

```
model2.predict(df_final[-100000:]).drop(['Label'], axis=1).values)
```

Out[194]:

```
array([[1.02652742],  
       [1.00823689],  
       [0.03793159],  
       ...,  
       [0.03336699],  
       [0.96753294],  
       [0.99070404]])
```

## 5. Representación resultados

A continuación se mostrarán las primeras líneas de la tabla con la limpieza de datos.

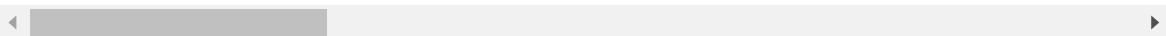
In [200]:

```
df_final.head()
```

Out[200]:

	DER_mass_MMC	DER_mass_transverse_met_lep	DER_mass_vis	DER_pt_h	DER_deltaeta_
1	160.937		68.768	103.235	48.146
3	143.905		81.417	80.943	0.414
4	175.864		16.915	134.805	16.405
5	89.744		13.550	59.149	116.344
8	105.594		50.559	100.989	4.288

5 rows × 26 columns



In [201]:

```
corr = df_final.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Out[201]:

	DER_mass_MMC	DER_mass_transverse_met_lep	DER_mass_v
DER_mass_MMC	1	-0.0231297	0.9121
DER_mass_transverse_met_lep	-0.0231297	1	-0.02571
DER_mass_vis	0.912168	-0.0257109	
DER_pt_h	0.0446253	-0.252249	-0.07758
DER_deltaeta_jet_jet	0.176749	-0.386791	0.1765
DER_mass_jet_jet	0.139955	-0.31845	0.1336
DER_prodeta_jet_jet	-0.194792	0.439589	-0.1767
DER_deltar_tau_lep	0.671695	-0.235891	0.6611
DER_pt_tot	0.0157431	-0.0494826	-0.01097
DER_sum_pt	0.267923	-0.170751	0.2021
DER_pt_ratio_lep_tau	-0.015224	0.46124	-0.03411
DER_met_phi_centrality	0.115169	-0.413174	-0.03158
DER_lep_eta_centrality	0.151104	-0.351869	0.1207
PRI_tau_pt	0.407592	-0.270367	0.4496
PRI_tau_eta	0.00909482	-0.00826645	0.006817
PRI_lep_pt	0.391858	0.391669	0.4400
PRI_lep_eta	0.00283515	-0.0096633	-0.0004458
PRI_met	-0.0163785	0.540391	-0.1931
PRI_met_sumet	0.145697	-0.158094	0.09401
PRI_jet_num	0.0318263	-0.213444	-0.05910
PRI_jet_leading_pt	0.0322604	-0.16457	-0.0609
PRI_jet_leading_eta	0.00630283	-0.00519143	0.00403
PRI_jet_subleading_pt	-0.036758	-0.194918	-0.1055
PRI_jet_subleading_eta	0.0109203	-0.00362624	0.007497
PRI_jet_all_pt	0.0376654	-0.239691	-0.0705
Label	-0.245822	0.407945	-0.2094

En la tabla superior se pueden ver las correlaciones de todas las variables, no solo con la columna Label.

## 6. Resolución del problema

Con los resultados obtenidos se puede concluir que las correlaciones entre las variables y la columna Label no son muy altas, pero al tener un gran número de ellas y tener influencia la variable Label en una gran cantidad de ellas hace que el modelo de predicción sea bastante exacto, pudiendo determinar si una partícula es o no bosón de Higgs con un alto porcentaje de acierto, esto quiere decir que se ha podido resolver el problema al que nos enfrentábamos.

## Datos de la práctica

Práctica realizada por Emilio José Lucas Marcos (ejlm)

Contribuciones	Firma
Investigación previa	ejlm
Redacción de las respuestas	ejlm
Desarrollo código	ejlm

In [ ]: