Nasir Islam Sujan Follow
Polyglot programmer 💻 Machine learning enthusiast | UI/UX designer. Currently pursuing my Bsc. in Software Engineering at Daffodil International University.
Sep 3 · 5 min read

An Exploratory Data Analysis on Lower Back Pain

Lower back pain, also called **lumbago**, is not a disorder. It's a symptom of several different types of medical problems. It usually results from a problem with one or more parts of the lower back, such as:

- ligaments
- muscles
- nerves
- the bony structures that make up the spine, called vertebral bodies or vertebrae

It can also be due to a problem with nearby organs, such as the kidneys.

. . .

In this EDA, I am going to use the Lower Back Pain Symptoms Dataset and try to find out interesting insights of this dataset. Lets begin!

Dataset Description

The dataset contains:

- **310** Observations
- **12** Features
- **1** Label

Search this file…

col. no.	Attribute name	type
Col1	pelvic_incidence	float64
Col2	pelvic_tilt	float64
Col3	lumbar_lordosis_angle	float64
Col4	sacral_slope	float64
Col5	pelvic_radius	float64
Col6	degree_spondylolisthesis	float64
Col7	pelvic_slope	float64
Col8	Direct_tilt	float64
Col9	thoracic_slope	float64

Importing Necessary Packages:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline


import seaborn as sns
sns.set()
from sklearn.preprocessing import MinMaxScaler,
StandardScaler, LabelEncoder
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier, plot_importance
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix
```

Reading the `.csv` file:

```
dataset = pd.read_csv("../input/Dataset_spine.csv")
```

Viewing the top 5 rows from the dataset:

```
dataset.head() # this will return top 5 rows
```

Removing dummy column:

```
# This command will remove the last column from our dataset.
del dataset["Unnamed: 13"]
```

Dataset Summary:

DataFrame.describe() method Generates a descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding `NaN` values. This method tells us a lot of things about a dataset. One important thing is that the `describe()` method deals only with numeric values. It doesn't work with any categorical values.

Now, let's understand the statistics that are generated by the `describe()` method:

- `Count` tells us the number of `NoN-empty` rows in a feature.
- `mean` tells us the mean value of that feature.
- `std` tells us the Standard Deviation Value of that feature.
- `min` tells us the minimum value of that feature.
- `25%`, `50%`, and `75%` are the percentile/quartile of each features. This quartile information helps us to detect Outliers.
- `max` tells us the maximum value of that feature.

```
dataset.describe()
```

	Col1	Col2	Col3	
count	310.000000	310.000000	310.000000	
mean	60.496653	17.542822	51.930930	
std	17.236520	10.008330	18.554064	
min	26.147921	-6.554948	14.000000	
25%	46.430294	10.667069	37.000000	
50%	58.691038	16.357689	49.562398	

dataset.describe() method output

Rename column for increase readability:

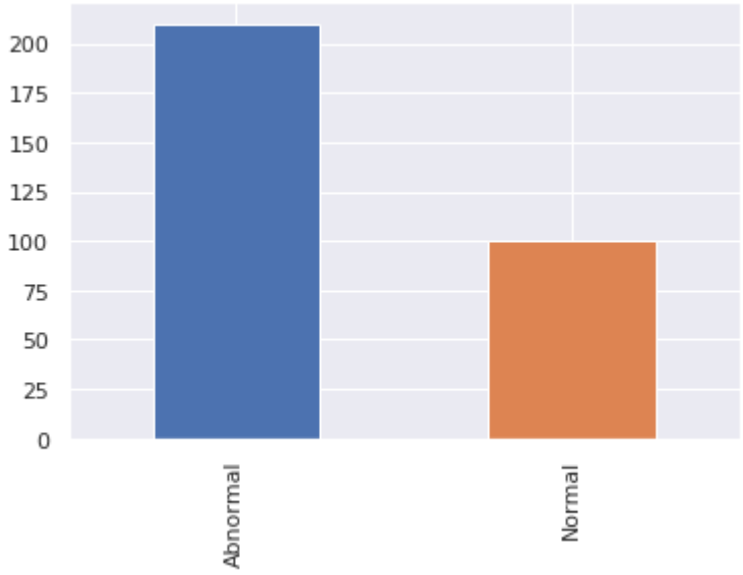
```
dataset.rename(columns = {
    "Col1" : "pelvic_incidence",
    "Col2" : "pelvic_tilt",
    "Col3" : "lumbar_lordosis_angle",
    "Col4" : "sacral_slope",
    "Col5" : "pelvic_radius",
    "Col6" : "degree_spondylolisthesis",
    "Col7" : "pelvic_slope",
    "Col8" : "direct_tilt",
    "Col9" : "thoracic_slope",
    "Col10" :"cervical_tilt",
    "Col11" : "sacrum_angle",
    "Col12" : "scoliosis_slope",
    "Class_att" : "class"}, inplace=True)
```

DataFrame.info() prints information about a DataFrame including the `index` dtype and `column` dtypes, `non-null` values and memory usage. We can use the `info()` to know whether a dataset contains any missing value or not.

```
dataset.info()
```

Visualize the number of abnormal and normal cases:

```
dataset["class"].value_counts().sort_index().plot.bar()
```



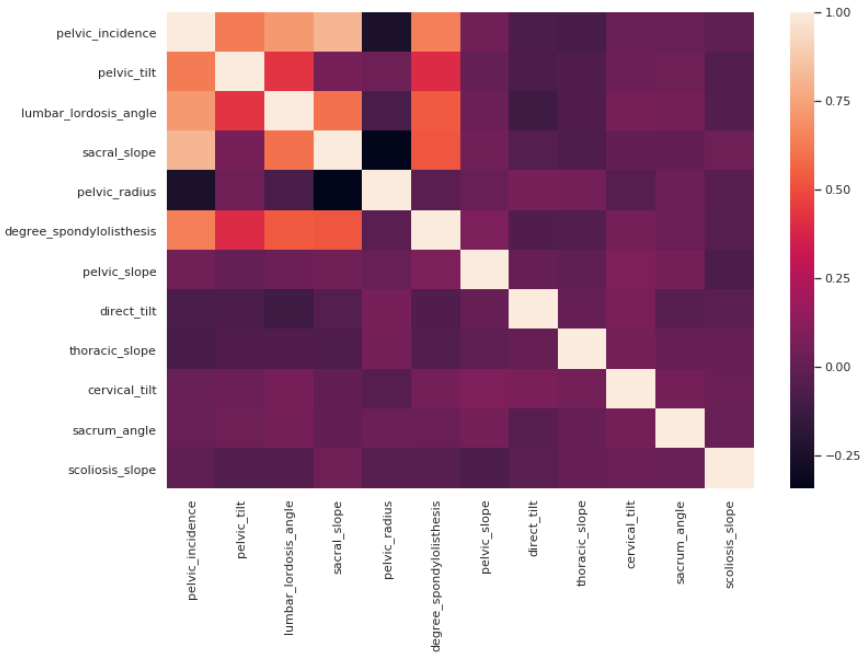
class distribution

Checking Correlation between features:

```
dataset.corr()
```

Visualize the correlation with heatmap:

```
plt.subplots(figsize=(12,8))
sns.heatmap(dataset.corr())
```



correlation between features

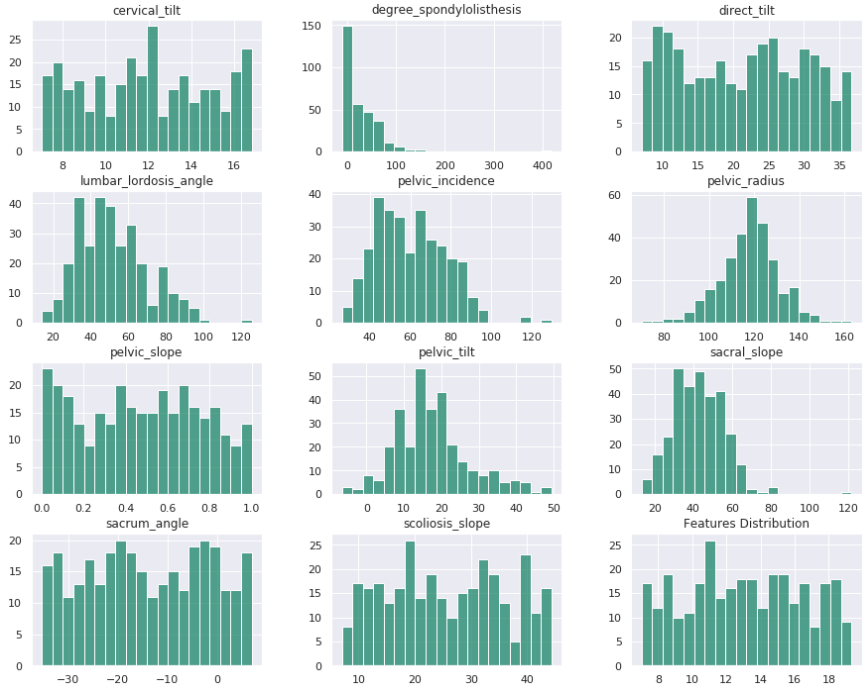
Custom correlogram:

```
sns.pairplot(dataset, hue="class")
```

custom correlogram

Visualize Features with Histogram:

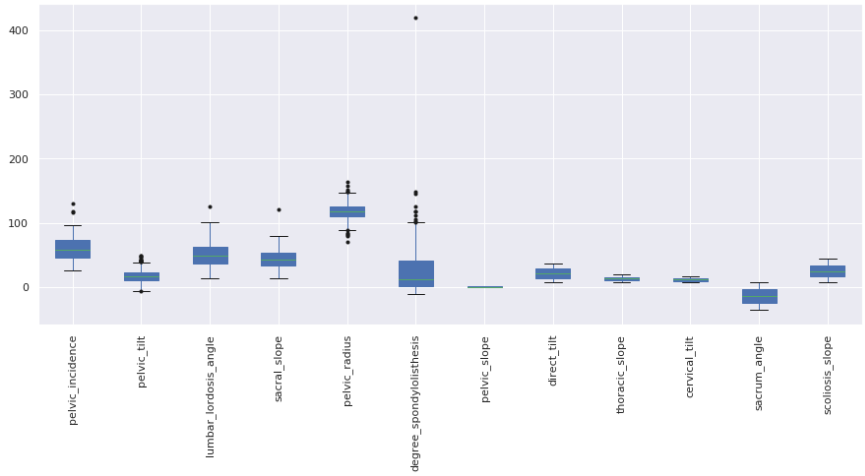
```
dataset.hist(figsize=(15,12),bins = 20, color="#007959AA")
plt.title("Features Distribution")
plt.show()
```



features histogram

Detecting and Removing Outliers

```
plt.subplots(figsize=(15,6))
dataset.boxplot(patch_artist=True, sym="k.")
plt.xticks(rotation=90)
```

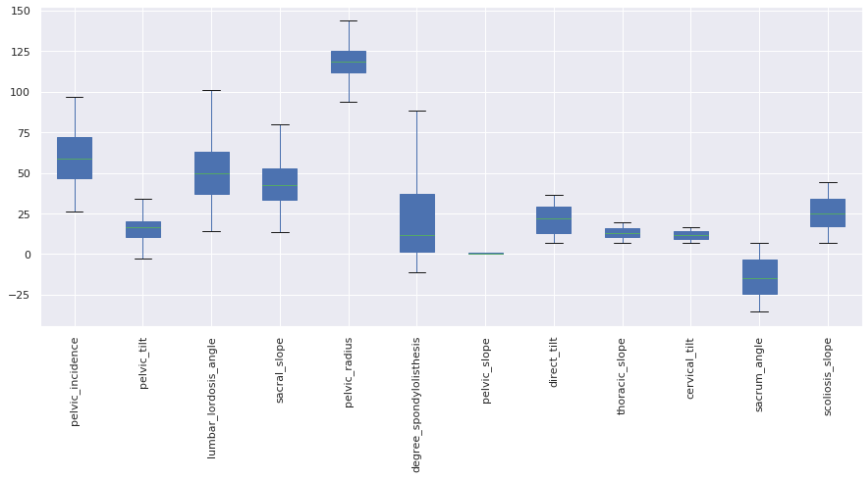


Detect outliers using boxplot

Remove Outliers:

```
def  remove_outlier(feature):
    first_q = np.percentile(X[feature], 25)
    third_q = np.percentile(X[feature], 75)
    IQR = third_q - first_q
    IQR *= 1.5
    minimum = first_q - IQR
    maximum = third_q + IQR

    mean = X[feature].median()

    X.loc[X[feature] < minimum, feature] = mean
    X.loc[X[feature] > maximum, feature] = mean


# taking all the columns except the last one
# last column is the label
X = dataset.iloc[:, :-1]


for i in range(len(X.columns)):
        remove_outlier(X.columns[i])
```

After removing Outliers:



features distribution after removing outliers

Feature Scaling:

Feature scaling though standardization (or Z-score normalization) can be an important preprocessing step for many machine learning algorithms. Our dataset contain features with highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Euclidean distance between two data points in their computations, this will create a problem. To avoid this effect, we need to bring all features to the same level of magnitudes. This can be achieved by feature scaling.

```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(X)
scaled_df = pd.DataFrame(data = scaled_data, columns =
X.columns)
scaled_df.head()
```

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_
0	0.523049	0.684601	0.295226
1	0.183082	0.349535	0.126987
2	0.605368	0.675639	0.416076
3	0.611962	0.740938	0.349432

dataset head after feature scaling

Label Encoding:

Certain algorithms like XGBoost can only have numerical values as their predictor variables. Hence we need encode our categorical values. LabelEncoder from `sklearn.preprocessing` package encode labels with value between `0` and `n_classes-1` .

```
label = dataset["class"]


encoder = LabelEncoder()
label = encoder.fit_transform(label)
```

. . .

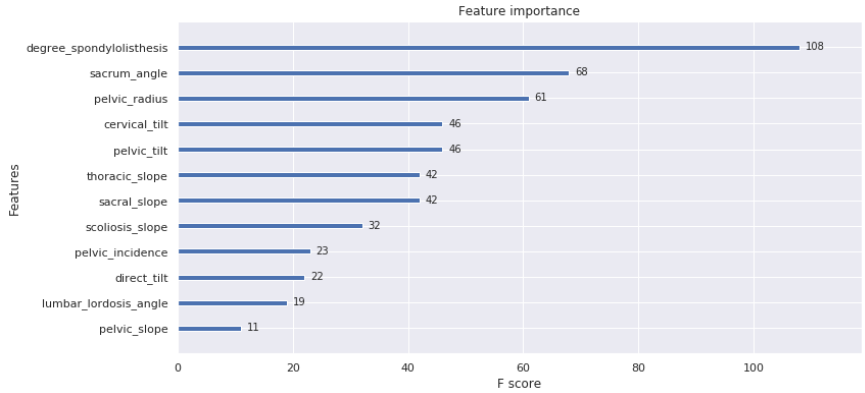
Model Training and Evaluation:

```
X = scaled_df
y = label


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15, random_state=0)


clf_gnb = GaussianNB()
pred_gnb = clf_gnb.fit(X_train, y_train).predict(X_test)
accuracy_score(pred_gnb, y_test)


# Out []: 0.8085106382978723
```

```
clf_svc = SVC(kernel="linear")
pred_svc = clf_svc.fit(X_train, y_train).predict(X_test)
accuracy_score(pred_svc, y_test)


# Out []: 0.7872340425531915


clf_xgb =  XGBClassifier()
pred_xgb = clf_xgb.fit(X_train, y_train).predict(X_test)
accuracy_score(pred_xgb, y_test)


# Out []: 0.8297872340425532
```

Feature Importance:

```
fig, ax = plt.subplots(figsize=(12, 6))
plot_importance(clf_xgb, ax=ax)
```



feature importance

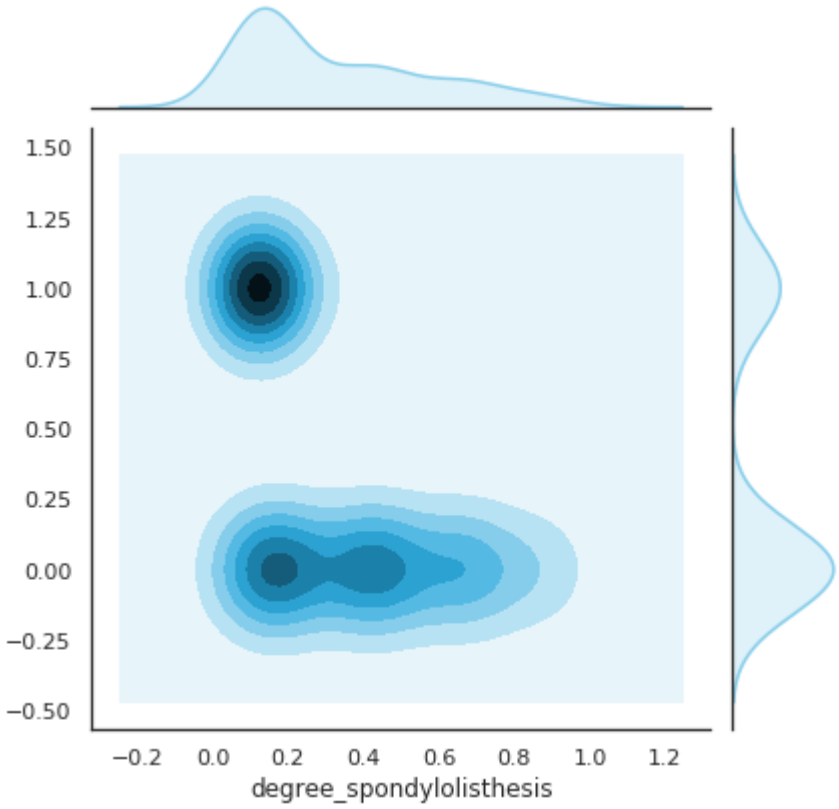
Marginal plot

A marginal plot allows to study the relationship between 2 numeric variables. The central chart display their correlation.

Lets visualize the relationship between `degree_spondylolisthesis` and `class` :

```
sns.set(style="white", color_codes=True)
sns.jointplot(x=X["degree_spondylolisthesis"], y=label,
```

```
  kind='kde', color="skyblue")
```



Marginal plot between degree_spondylolisthesis and class

Thats All. Thanks for reading. :)

For the full code visit Kaggle or Google Colab.

If you like this article then give 👏 clap. Happy Coding!