Visualization Python Cheat sheet

Table of Contents

[Common Section 2](#_Toc524097604)

[Matplotlib Library 2](#_Toc524097605)

[1. Line Plot 2](#_Toc524097606)

[2. Subplots 2](#_Toc524097607)

[3. Learning and Validation Plots with Matplotlib and sklearn library 4](#_Toc524097608)

[4. 3D Matplotlib 5](#_Toc524097609)

[Seaborn Library 6](#_Toc524097610)

[1. Heatmap 6](#_Toc524097611)

[2. Scatter Plots 7](#_Toc524097613)

[3. Swarm Plots 8](#_Toc524097614)

[4. Box Plots 9](#_Toc524097615)

[5. Pair Plot 10](#_Toc524097616)

# **Common Section**

dataset = pd.read\_csv(‘Name\_of\_the\_csv\_file.csv’)

name\_of\_cols = list(dataset)[1:-1] #Name of Feature Columns

name\_output\_col = ‘class’

# **Matplotlib Library**

import matplotlib.pyplot as plt

%matplotlib inline

## **Line Plot**

plt.figure(figsize=(25, 5))

plt.plot(x\_df, y\_df, color='red', linestyle='dashed', marker='o', markerfacecolor='blue', markersize=10)

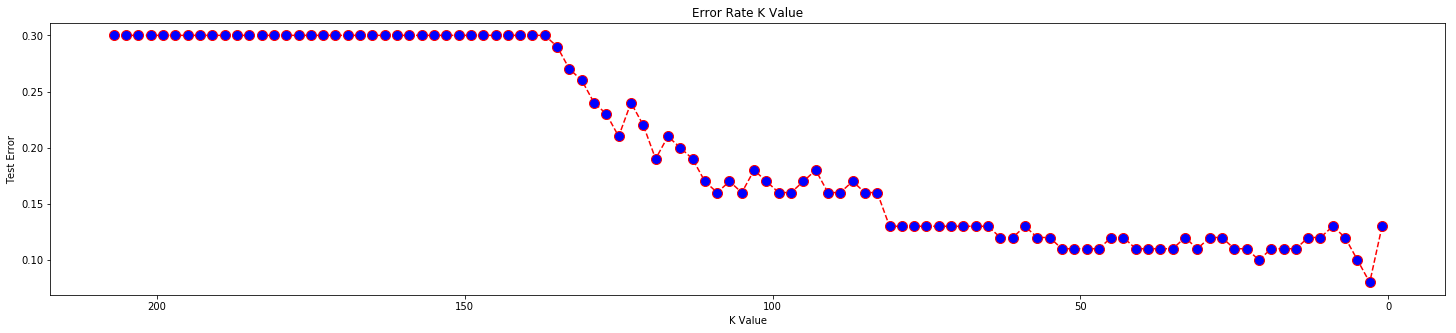
plt.title('Error Rate K Value')

plt.xlabel('K Value')

plt.ylabel('Test Error')

plt.gca().invert\_xaxis() #This is to invert the X-Axis

plt.show()



## **Subplots**

fig, axes = plt.subplots(2, 1, figsize=(25, 20), squeeze=False)

#Axes – 0 row 0 column

axes[0,0].plot(val, results['mean\_train\_AUC'], label='AUC ROC SCORE',color='g', marker='o', linestyle='dashed',linewidth=2, markersize=12)

axes[0,0].plot(val, results['mean\_train\_Accuracy'], label='ACCURACY SCORE',color='r', marker='+', linestyle='dashed',linewidth=2, markersize=12)

axes[0,0].plot(val, results['mean\_train\_Precision'], label='PRECISION SCORE',color='b', marker='s', linestyle='dashed', linewidth=2, markersize=12)

axes[0,0].plot(val, results['mean\_train\_Recall'], label='RECALL SCORE',color='k', marker='x', linestyle='dashed',linewidth=2, markersize=12)

axes[0,0].set\_xlabel('K Parameter', fontsize=25)

axes[0,0].set\_ylabel('Performance', fontsize=25)

axes[0,0].set\_title('TRAIN SCORES', fontsize=30)

axes[0,0].legend(loc='lower right', prop={'size': 20})

#Axes – 1 Row 0 Column

axes[1,0].plot(val, results['mean\_test\_AUC'], label='AUC ROC SCORE',color='g', marker='o', linestyle='dashed', linewidth=2, markersize=12)

axes[1,0].plot(val, results['mean\_test\_Accuracy'], label='ACCURACY SCORE',color='r', marker='+', linestyle='dashed', linewidth=2, markersize=12)

axes[1,0].plot(val, results['mean\_test\_Precision'], label='PRECISION SCORE',color='b', marker='s', linestyle='dashed', linewidth=2, markersize=12)

axes[1,0].plot(val, results['mean\_test\_Recall'], label='RECALL SCORE',color='k', marker='x', linestyle='dashed', linewidth=2, markersize=12)

axes[1,0].set\_xlabel('K Parameter', fontsize=25)

axes[1,0].set\_ylabel('Performance', fontsize=25)

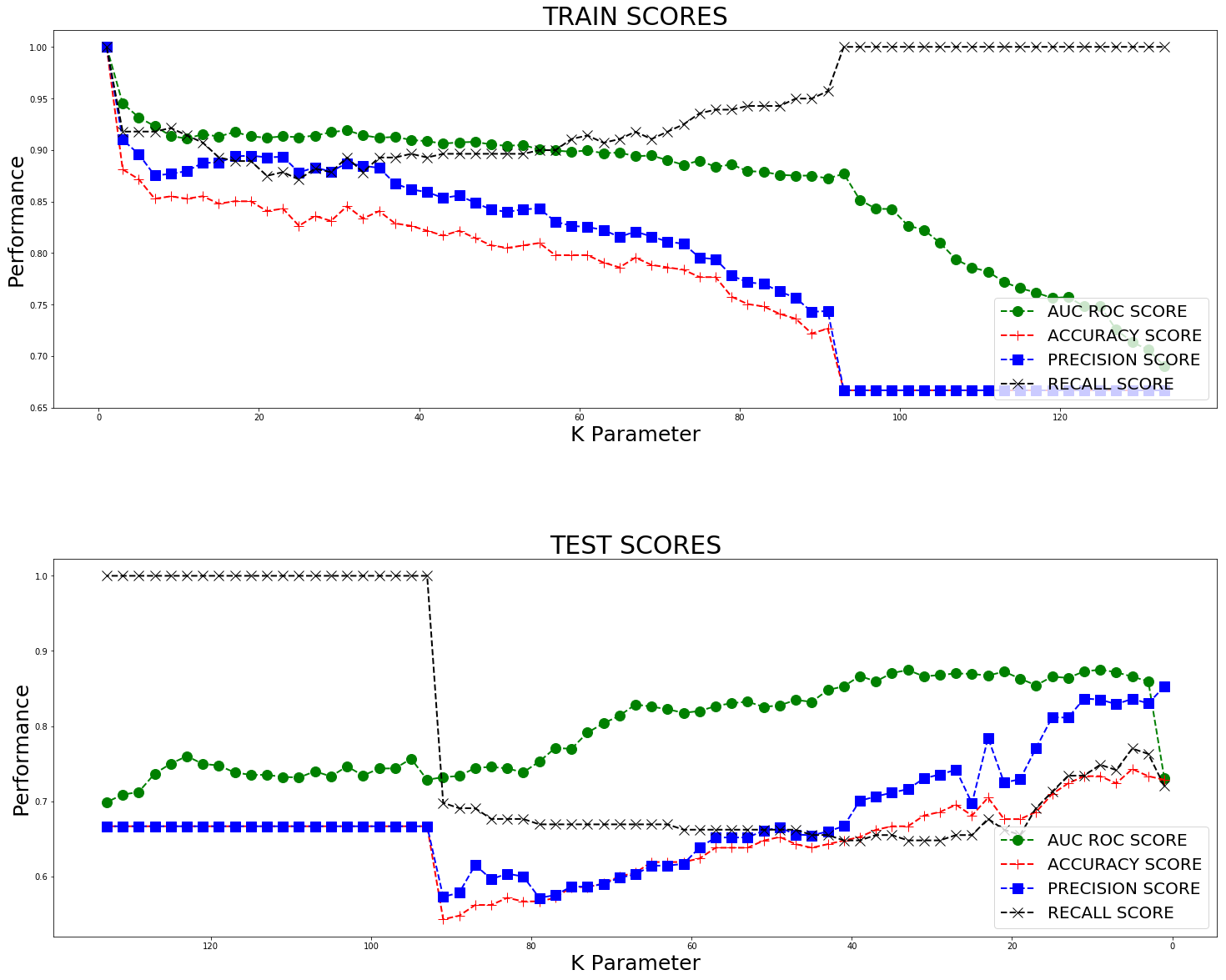
axes[1,0].set\_title('TEST SCORES', fontsize=30)

axes[1,0].legend(loc='lower right', prop={'size': 20})

plt.gca().invert\_xaxis()

fig.subplots\_adjust(hspace=0.4)

plt.show()



## **Learning and Validation Plots with Matplotlib and sklearn library**

#Making Learning Curve using Sklearn Library

from sklearn.model\_selection import learning\_curve

#Cross Validation

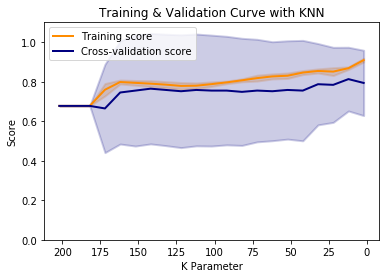
from sklearn.model\_selection import validation\_curve

paramr = list(range(2,210,10))

train\_scores, test\_scores = validation\_curve(

KNeighborsClassifier(), dataset.iloc[:, 1:-1], dataset.iloc[:, -1], param\_name="n\_neighbors",

param\_range=paramr, cv=10, scoring="accuracy", n\_jobs=1)

train\_scores\_mean = np.mean(train\_scores, axis=1)

train\_scores\_std = np.std(train\_scores, axis=1)

test\_scores\_mean = np.mean(test\_scores, axis=1)

test\_scores\_std = np.std(test\_scores, axis=1)

plt.title("Training & Validation Curve with KNN")

plt.xlabel("K Parameter")

plt.ylabel("Score")

plt.ylim(0.0, 1.1)

lw = 2

plt.plot(paramr, train\_scores\_mean, label="Training score", color="darkorange", lw=lw)

plt.fill\_between(paramr, train\_scores\_mean - train\_scores\_std, train\_scores\_mean + train\_scores\_std, alpha=0.2,

color="darkorange", lw=lw)

plt.plot(paramr, test\_scores\_mean, label="Cross-validation score", color="navy", lw=lw)

plt.fill\_between(paramr, test\_scores\_mean - test\_scores\_std, test\_scores\_mean + test\_scores\_std, alpha=0.2,

color="navy", lw=lw)

plt.legend(loc="best")

plt.gca().invert\_xaxis()

plt.show()

## **3D Matplotlib**

import matplotlib as mpl

from mpl\_toolkits.mplot3d import Axes3D

mpl.rcParams['legend.fontsize'] = 10

#3D plot with all three parameters i.e. Training size N vs K Parameter vs Test Errors

fig = plt.figure(figsize=(10,7))

ax = fig.gca(projection='3d')

ax.plot(x, y, z, label='learning Curve',color='green', marker='o')

ax.legend(prop={'size':12})

ax.set\_xlabel('Train Size', fontsize=12)

ax.set\_ylabel('Optimal K Parameter', fontsize=12)

ax.set\_zlabel('Best Test Errors', fontsize=12)

ax.set\_title('Training size vs K Parameter vs Test Errors', fontsize=15)

plt.show()



# **Seaborn Library**

Import seaborn as sns

## **Heatmap**

#Plotting a heatmap showing correlation between features

corr = dataset.iloc[:,1:-1].corr()

# Generate a mask for the upper triangle

mask = np.zeros\_like(corr, dtype=np.bool)

mask[np.triu\_indices\_from(mask)] = True

# Set up the matplotlib figure

f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap

cmap = sb.diverging\_palette(220, 10, as\_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio

sb.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0, square=True, linewidths=.5, cbar\_kws={"shrink": .5})

## 

## **Scatter Plots**

#Make scatterplots of the independent variables in the dataset. Use color to show Classes 0 and 1.

fig, axes = plt.subplots(2,3, figsize=(10,7))

ind = 0

#For Left Image – without much formatting

for i in range(2):

for j in range(3):

sb.scatterplot(x=name\_output\_col, y=name\_of\_cols[ind], data=dataset, hue='class', ax=axes[i,j])

ind = ind + 1

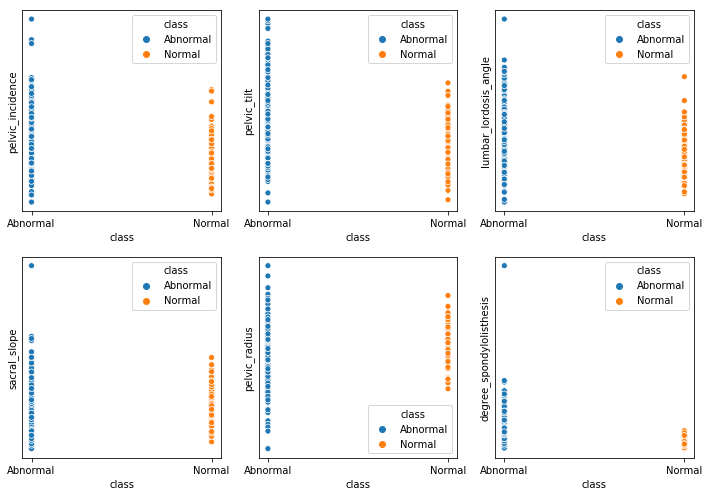
#style parameter: name of one column. It gives different markers to different features based on column passed.

#palette parameter: sb.color\_palette("Set1", n\_colors=’Number of Labels’) – General format to set palette

#For more palette option visit- #https://seaborn.pydata.org/generated/seaborn.color\_palette.html#seaborn.color\_palette

plt.setp(axes, yticks=[])

plt.tight\_layout()



## **Swarm Plots**

#Make scatterplots of the independent variables in the dataset. Use color to show Classes 0 and 1.

fig, axes = plt.subplots(2,3, figsize=(10,7), sharey=True)

ind = 0

for i in range(2):

for j in range(3):

sb.swarmplot(x=name\_of\_cols[ind], y=name\_output\_col, data=dataset, hue='class',

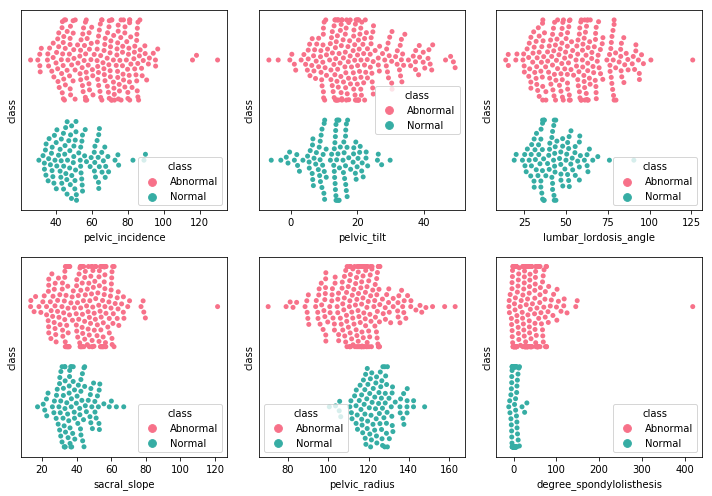
palette=sb.color\_palette("husl", n\_colors=2),ax=axes[i,j])

ind = ind + 1

#Similar to Scatter plot, you can choose your own color palette

plt.setp(axes, yticks=[])

plt.tight\_layout()



## **Box Plots**

#Make boxplots for each of the independent variables. Use color to show Classes 0 and 1

fig, axes = plt.subplots(2,3, figsize=(12,5), sharey=True)

ind = 0

for i in range(2):

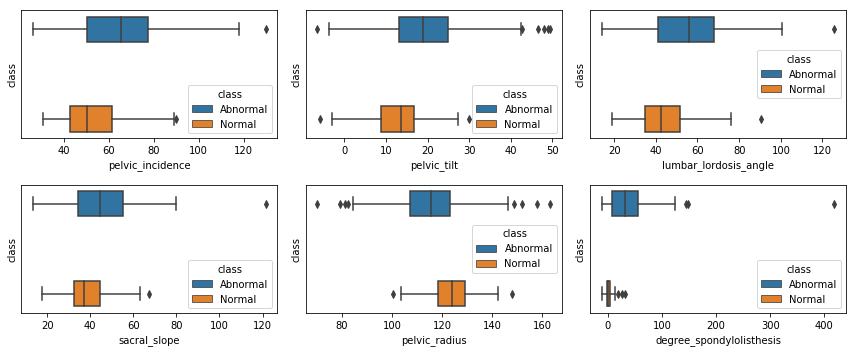
for j in range(3):

sb.boxplot(x=name\_of\_cols[ind], y=name\_output\_col, data=dataset, hue='class', ax=axes[i,j])

ind = ind + 1

plt.setp(axes, yticks=[])

plt.tight\_layout()



## **Pair Plot**

#The most AMAZING plot

sb.pairplot(dataset,hue='class',palette=sb.color\_palette("muted", n\_colors=2))

