**How to visualize your Machine Learning projects in Python?**

Machine Learning visualization can help you display the models intuitively and understand them. In this article, I am going to introduce you a package to draw your ML projects — yellowbrick.

We need to install it before using:

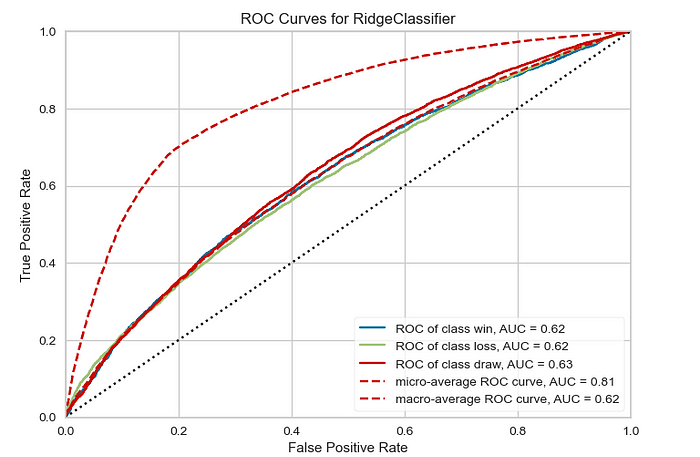
!pip install yellowbrick

Then import the packages we need:

import matplotlib.pyplot as plt  
plt.figure(dpi=120)  
  
from sklearn.linear\_model import RidgeClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder  
  
from yellowbrick.classifier import ROCAUC  
from yellowbrick.datasets import load\_game  
  
# yellowbrick relies on sklearn and matplotlib

1. ROC Curve

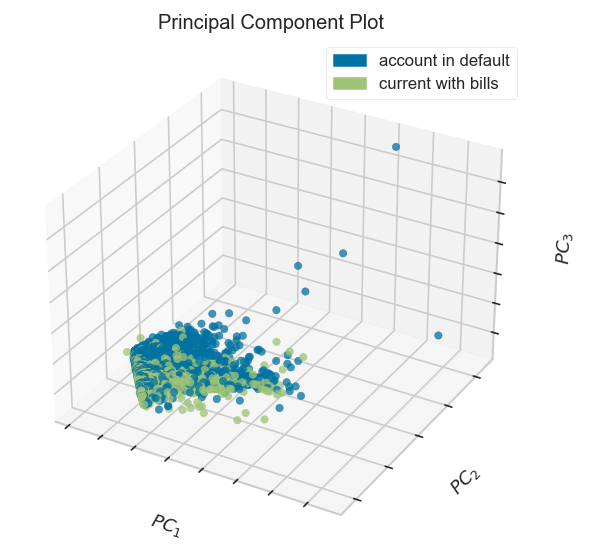
# Load data  
X, y = load\_game()  
  
# Transformation  
X = OrdinalEncoder().fit\_transform(X)  
y = LabelEncoder().fit\_transform(y)  
  
# Split data  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42)  
  
# Build a model and draw the ROC  
model = RidgeClassifier()  
visualizer = ROCAUC(model, classes=["win", "loss", "draw"])  
  
visualizer.fit(X\_train, y\_train)   
visualizer.score(X\_test, y\_test) # Evaluation  
visualizer.show()



In just a few lines of code, it gives you ROC curves by classes with specific models.

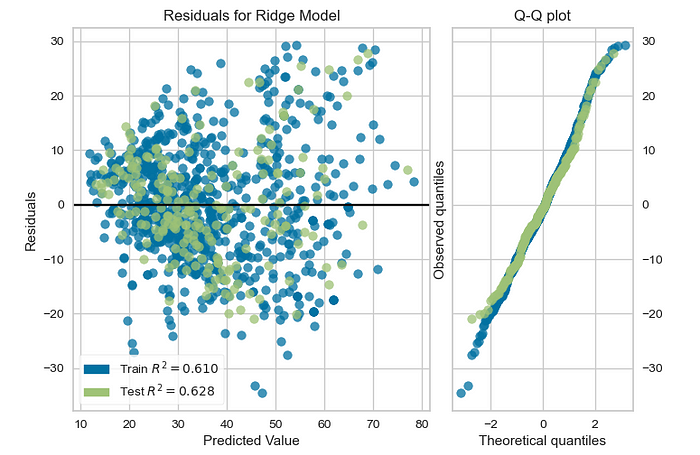
2. PCA

plt.figure(dpi=120)  
  
from yellowbrick.features import PCA  
from yellowbrick.datasets import load\_credit  
  
X, y = load\_credit()  
classes = ['account in default', 'current with bills']  
  
visualizer = PCA(scale=True, projection=3, classes=classes)  
visualizer.fit\_transform(X, y)  
visualizer.show()



3. Q-Q plot for regression models

from sklearn.linear\_model import Ridge  
  
from yellowbrick.datasets import load\_concrete  
from yellowbrick.regressor import ResidualsPlot  
  
X, y = load\_concrete()  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
model = Ridge()  
visualizer = ResidualsPlot(model, hist=False, qqplot=True)  
visualizer.fit(X\_train, y\_train)  
visualizer.score(X\_test, y\_test)  
visualizer.show()



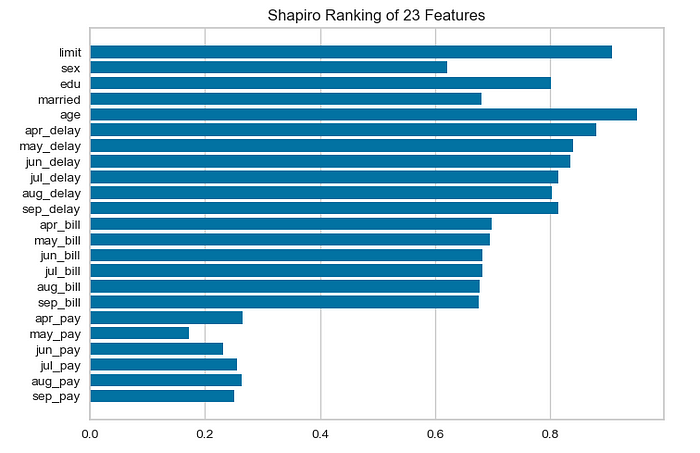
4. Lasso

plt.figure(dpi=120)  
 from sklearn.linear\_model import Lasso  
 from yellowbrick.datasets import load\_bikeshare  
 from yellowbrick.regressor import prediction\_error  
   
  
X, y = load\_bikeshare()  
 visualizer = prediction\_error(Lasso(), X, y) # Only one line of code!

5. Feature Analysis

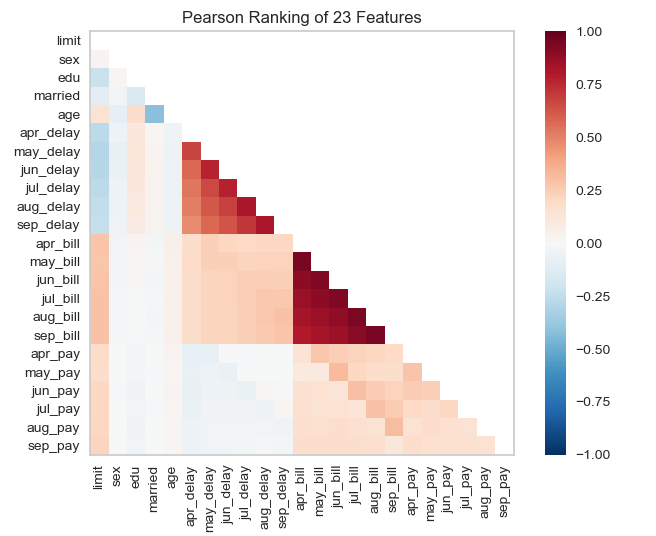
(1) Rank 1D

from yellowbrick.datasets import load\_credit  
from yellowbrick.features import Rank1D  
  
X, y = load\_credit()  
  
visualizer = Rank1D(algorithm='shapiro')  
  
visualizer.fit(X, y)   
visualizer.transform(X)   
visualizer.show()



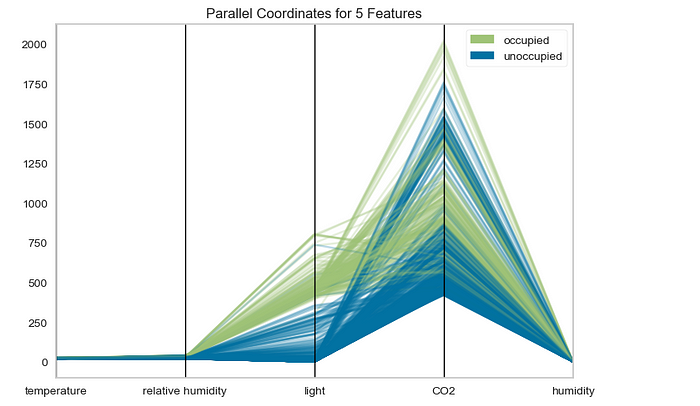
(2) Rank 2D

from yellowbrick.datasets import load\_credit  
from yellowbrick.features import Rank2D  
  
X, y = load\_credit()  
  
visualizer = Rank2D(algorithm='pearson')  
  
visualizer.fit(X, y)   
visualizer.transform(X)   
visualizer.show()



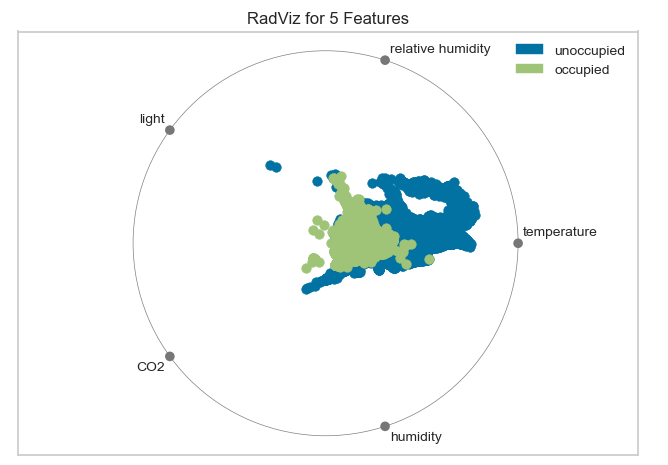
(3) Parallel Coordinates

from yellowbrick.features import ParallelCoordinates  
from yellowbrick.datasets import load\_occupancy  
  
X, y = load\_occupancy()  
  
# Specify the features of interest and the classes of the target  
features = [  
 "temperature", "relative humidity", "light", "CO2", "humidity"  
]  
classes = ["unoccupied", "occupied"]  
  
visualizer = ParallelCoordinates(  
 classes=classes, features=features, sample=0.05, shuffle=True  
)  
  
visualizer.fit\_transform(X, y)  
  
visualizer.show()



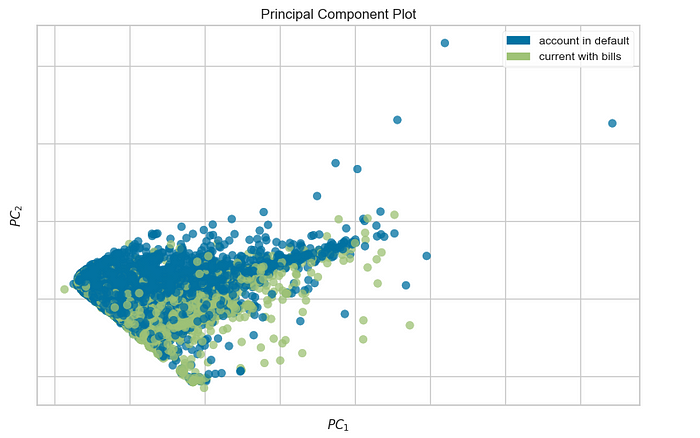
(4) RadViz

from yellowbrick.datasets import load\_occupancy  
from yellowbrick.features import RadViz  
  
X, y = load\_occupancy()  
  
# Specify the target classes  
classes = ["unoccupied", "occupied"]  
  
visualizer = RadViz(classes=classes)  
  
visualizer.fit(X, y)   
visualizer.transform(X)   
visualizer.show()



(5) PCA Projection

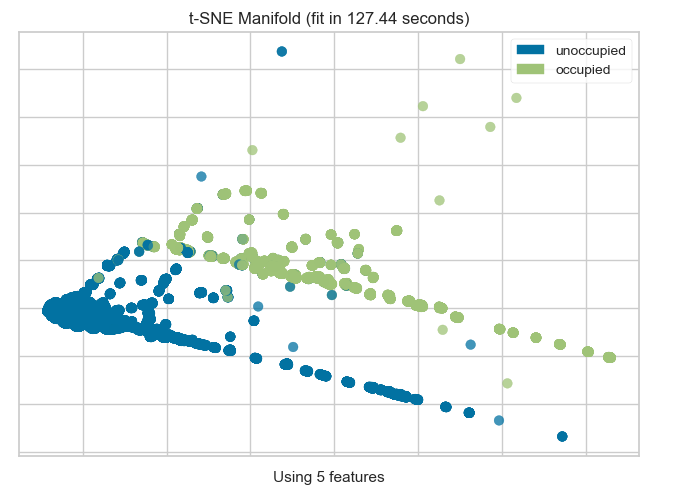
from yellowbrick.datasets import load\_credit  
from yellowbrick.features import PCA  
  
# Specify the features of interest and the target  
X, y = load\_credit()  
classes = ['account in default', 'current with bills']  
  
visualizer = PCA(scale=True, classes=classes)  
visualizer.fit\_transform(X, y)  
visualizer.show()



(6) Manifold

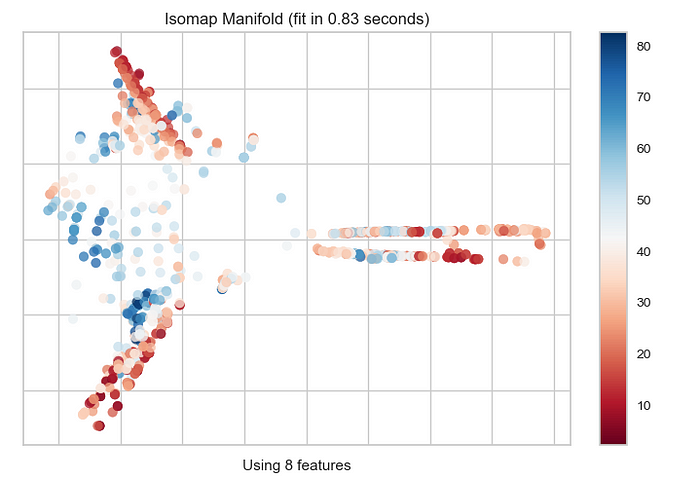
<1> Discrete Target

from yellowbrick.features import Manifold  
from yellowbrick.datasets import load\_occupancy  
  
X, y = load\_occupancy()  
classes = ["unoccupied", "occupied"]  
  
viz = Manifold(manifold="tsne", classes=classes)  
  
viz.fit\_transform(X, y)   
viz.show()



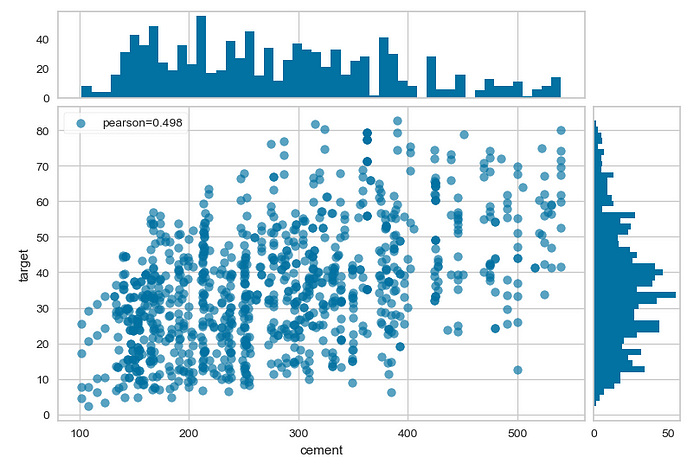
<2> Continuous Target

from yellowbrick.features import Manifold  
from yellowbrick.datasets import load\_concrete  
  
X, y = load\_concrete()  
  
viz = Manifold(manifold="isomap", n\_neighbors=10)  
  
viz.fit\_transform(X, y)   
viz.show()



(7) Joint plot

from yellowbrick.datasets import load\_concrete  
from yellowbrick.features import JointPlotVisualizer  
  
X, y = load\_concrete()  
  
visualizer = JointPlotVisualizer(columns="cement")  
  
visualizer.fit\_transform(X, y)   
visualizer.show()



Thank you for reading.

# Reference: <https://www.scikit-yb.org/en/latest/index.html> for more visuls.