# Week 8:

File: DSC550\_Paulovici\_Exercise\_8\_2.py (.ipynb)  
Name: Kevin Paulovici  
Date: 5/2/2020  
Course: DSC 550 Data Mining (2205-1)  
Assignment: 8.2 Exercise: Titanic Case Study Part 3

# Part 1

**Assignment Tasks**

Complete the Titanic Case Study Part 1 tutorial. This will be a complete Analysis Case study but Part 1 is the Graph Analysis. I have provided sample code for you to use as you go through the tutorial. I recommend that you comment out the steps and run them separately so you can fully understand what you are doing for each step of the analysis. As you go through each step, take screenshots to “prove” to me that you successfully completed each step. Paste your screenshots into a Word document and submit that Word document to the Assignment submission link.  
  
Code provided by Prof. Becky Deitenbeck

In [1]:

*#Titanic Tutorial Part 1*

*#Graphics Analysis*

**import** **pandas** **as** **pd**

**import** **yellowbrick**

In [2]:

*#Step 1: Load data into a dataframe*

addr1 = "train.csv"

data = pd.read\_csv(addr1)

In [3]:

*# Step 2: check the dimension of the table*

print("The dimension of the table is: ", data.shape)

The dimension of the table is: (891, 12)

In [4]:

*#Step 3: Look at the data*

print(data.head(5))

PassengerId Survived Pclass \

0 1 0 3

1 2 1 1

2 3 1 3

3 4 1 1

4 5 0 3

Name Sex Age SibSp \

0 Braund, Mr. Owen Harris male 22.0 1

1 Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1

2 Heikkinen, Miss. Laina female 26.0 0

3 Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1

4 Allen, Mr. William Henry male 35.0 0

Parch Ticket Fare Cabin Embarked

0 0 A/5 21171 7.2500 NaN S

1 0 PC 17599 71.2833 C85 C

2 0 STON/O2. 3101282 7.9250 NaN S

3 0 113803 53.1000 C123 S

4 0 373450 8.0500 NaN S

In [5]:

*#Step 5: what type of variables are in the table*

print("Describe Data")

print(data.describe())

print("Summarized Data")

print(data.describe(include=['O']))

Describe Data

PassengerId Survived Pclass Age SibSp \

count 891.000000 891.000000 891.000000 714.000000 891.000000

mean 446.000000 0.383838 2.308642 29.699118 0.523008

std 257.353842 0.486592 0.836071 14.526497 1.102743

min 1.000000 0.000000 1.000000 0.420000 0.000000

25% 223.500000 0.000000 2.000000 20.125000 0.000000

50% 446.000000 0.000000 3.000000 28.000000 0.000000

75% 668.500000 1.000000 3.000000 38.000000 1.000000

max 891.000000 1.000000 3.000000 80.000000 8.000000

Parch Fare

count 891.000000 891.000000

mean 0.381594 32.204208

std 0.806057 49.693429

min 0.000000 0.000000

25% 0.000000 7.910400

50% 0.000000 14.454200

75% 0.000000 31.000000

max 6.000000 512.329200

Summarized Data

Name Sex Ticket Cabin Embarked

count 891 891 891 204 889

unique 891 2 681 147 3

top Beavan, Mr. William Thomas male 1601 G6 S

freq 1 577 7 4 644

In [6]:

*#Step 6: import visulization packages*

**import** **matplotlib.pyplot** **as** **plt**

*# set up the figure size*

plt.rcParams['figure.figsize'] = (20, 10)

*# make subplots*

fig, axes = plt.subplots(nrows = 2, ncols = 2)

*# Specify the features of interest*

num\_features = ['Age', 'SibSp', 'Parch', 'Fare']

xaxes = num\_features

yaxes = ['Counts', 'Counts', 'Counts', 'Counts']

*# draw histograms*

axes = axes.ravel()

**for** idx, ax **in** enumerate(axes):

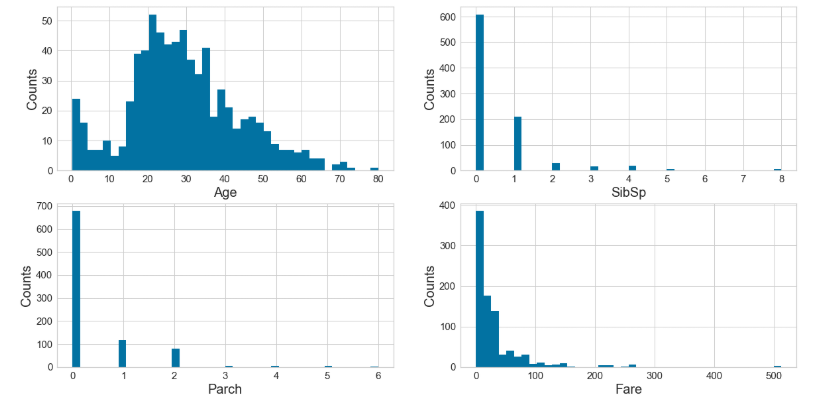
ax.hist(data[num\_features[idx]].dropna(), bins=40)

ax.set\_xlabel(xaxes[idx], fontsize=20)

ax.set\_ylabel(yaxes[idx], fontsize=20)

ax.tick\_params(axis='both', labelsize=15)

plt.show()



In [7]:

*#7: Barcharts: set up the figure size*

*#%matplotlib inline*

plt.rcParams['figure.figsize'] = (20, 10)

*# make subplots*

fig, axes = plt.subplots(nrows = 2, ncols = 2)

*# make the data read to feed into the visulizer*

X\_Survived = data.replace({'Survived': {1: 'yes', 0: 'no'}}).groupby('Survived').size().reset\_index(name='Counts')['Survived']

Y\_Survived = data.replace({'Survived': {1: 'yes', 0: 'no'}}).groupby('Survived').size().reset\_index(name='Counts')['Counts']

*# make the bar plot*

axes[0, 0].bar(X\_Survived, Y\_Survived)

axes[0, 0].set\_title('Survived', fontsize=25)

axes[0, 0].set\_ylabel('Counts', fontsize=20)

axes[0, 0].tick\_params(axis='both', labelsize=15)

*# make the data read to feed into the visulizer*

X\_Pclass = data.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}}).groupby('Pclass').size().reset\_index(name='Counts')['Pclass']

Y\_Pclass = data.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}}).groupby('Pclass').size().reset\_index(name='Counts')['Counts']

*# make the bar plot*

axes[0, 1].bar(X\_Pclass, Y\_Pclass)

axes[0, 1].set\_title('Pclass', fontsize=25)

axes[0, 1].set\_ylabel('Counts', fontsize=20)

axes[0, 1].tick\_params(axis='both', labelsize=15)

*# make the data read to feed into the visulizer*

X\_Sex = data.groupby('Sex').size().reset\_index(name='Counts')['Sex']

Y\_Sex = data.groupby('Sex').size().reset\_index(name='Counts')['Counts']

*# make the bar plot*

axes[1, 0].bar(X\_Sex, Y\_Sex)

axes[1, 0].set\_title('Sex', fontsize=25)

axes[1, 0].set\_ylabel('Counts', fontsize=20)

axes[1, 0].tick\_params(axis='both', labelsize=15)

*# make the data read to feed into the visulizer*

X\_Embarked = data.groupby('Embarked').size().reset\_index(name='Counts')['Embarked']

Y\_Embarked = data.groupby('Embarked').size().reset\_index(name='Counts')['Counts']

*# make the bar plot*

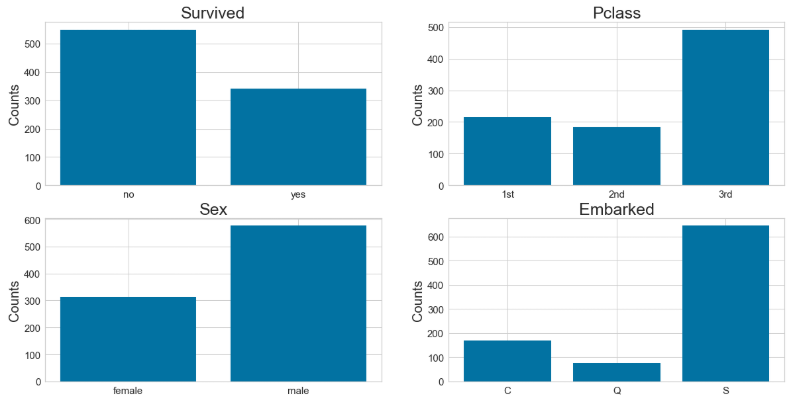
axes[1, 1].bar(X\_Embarked, Y\_Embarked)

axes[1, 1].set\_title('Embarked', fontsize=25)

axes[1, 1].set\_ylabel('Counts', fontsize=20)

axes[1, 1].tick\_params(axis='both', labelsize=15)

plt.show()



In [8]:

*#Step 8: Pearson Ranking*

*#set up the figure size*

*#%matplotlib inline*

plt.rcParams['figure.figsize'] = (15, 7)

*# import the package for visulization of the correlation*

**from** **yellowbrick.features** **import** Rank2D

*# extract the numpy arrays from the data frame*

X = data[num\_features].values

*# instantiate the visualizer with the Covariance ranking algorithm*

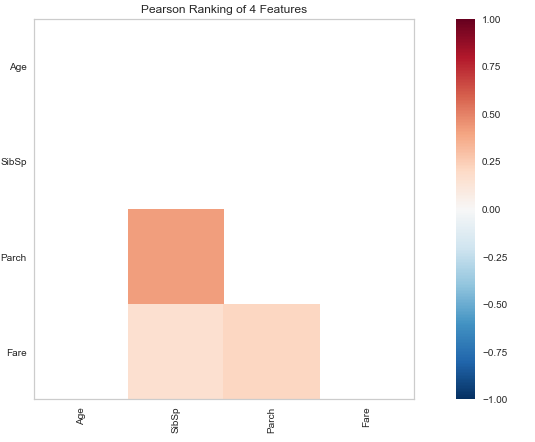
visualizer = Rank2D(features=num\_features, algorithm='pearson')

visualizer.fit(X) *# Fit the data to the visualizer*

visualizer.transform(X) *# Transform the data*

visualizer.poof(outpath="pcoords1.png") *# Draw/show/poof the data*

plt.show()



In [9]:

*# Step 9: Compare variables against Survived and Not Survived*

*#set up the figure size*

*#%matplotlib inline*

plt.rcParams['figure.figsize'] = (15, 7)

plt.rcParams['font.size'] = 50

*# setup the color for yellowbrick visulizer*

**from** **yellowbrick.style** **import** set\_palette

set\_palette('sns\_bright')

*# import packages*

**from** **yellowbrick.features** **import** ParallelCoordinates

*# Specify the features of interest and the classes of the target*

classes = ['Not-survived', 'Survived']

num\_features = ['Age', 'SibSp', 'Parch', 'Fare']

*# copy data to a new dataframe*

data\_norm = data.copy()

*# normalize data to 0-1 range*

**for** feature **in** num\_features:

data\_norm[feature] = (data[feature] - data[feature].mean(skipna=**True**)) / (data[feature].max(skipna=**True**) - data[feature].min(skipna=**True**))

*# Extract the numpy arrays from the data frame*

X = data\_norm[num\_features].values

y = data.Survived.values

*# Instantiate the visualizer*

*# Instantiate the visualizer*

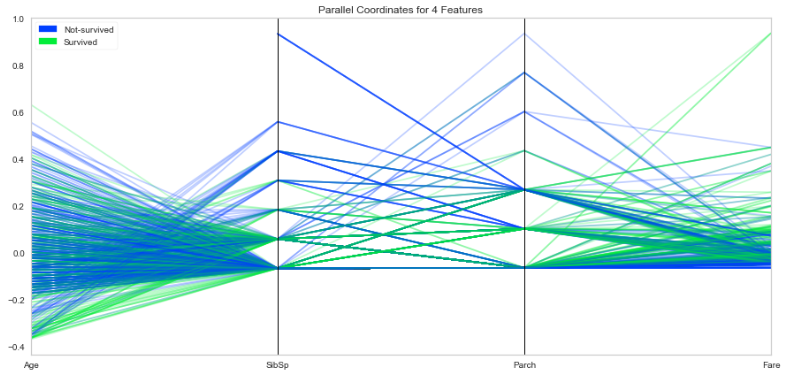
visualizer = ParallelCoordinates(classes=classes, features=num\_features)

visualizer.fit(X, y) *# Fit the data to the visualizer*

visualizer.transform(X) *# Transform the data*

visualizer.poof(outpath="pcoords2.png") *# Draw/show/poof the data*

plt.show()



In [10]:

*# Step 10 - stacked bar charts to compare survived/not survived*

*#set up the figure size*

*#%matplotlib inline*

plt.rcParams['figure.figsize'] = (20, 10)

*# make subplots*

fig, axes = plt.subplots(nrows = 2, ncols = 2)

*# make the data read to feed into the visulizer*

Sex\_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}})[data['Survived']==1]['Sex'].value\_counts()

Sex\_not\_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}})[data['Survived']==0]['Sex'].value\_counts()

Sex\_not\_survived = Sex\_not\_survived.reindex(index = Sex\_survived.index)

*# make the bar plot*

p1 = axes[0, 0].bar(Sex\_survived.index, Sex\_survived.values)

p2 = axes[0, 0].bar(Sex\_not\_survived.index, Sex\_not\_survived.values, bottom=Sex\_survived.values)

axes[0, 0].set\_title('Sex', fontsize=25)

axes[0, 0].set\_ylabel('Counts', fontsize=20)

axes[0, 0].tick\_params(axis='both', labelsize=15)

axes[0, 0].legend((p1[0], p2[0]), ('Survived', 'Not-survived'), fontsize = 15)

*# make the data read to feed into the visualizer*

Pclass\_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}}).replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}})[data['Survived']==1]['Pclass'].value\_counts()

Pclass\_not\_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}}).replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}})[data['Survived']==0]['Pclass'].value\_counts()

Pclass\_not\_survived = Pclass\_not\_survived.reindex(index = Pclass\_survived.index)

*# make the bar plot*

p3 = axes[0, 1].bar(Pclass\_survived.index, Pclass\_survived.values)

p4 = axes[0, 1].bar(Pclass\_not\_survived.index, Pclass\_not\_survived.values, bottom=Pclass\_survived.values)

axes[0, 1].set\_title('Pclass', fontsize=25)

axes[0, 1].set\_ylabel('Counts', fontsize=20)

axes[0, 1].tick\_params(axis='both', labelsize=15)

axes[0, 1].legend((p3[0], p4[0]), ('Survived', 'Not-survived'), fontsize = 15)

*# make the data read to feed into the visualizer*

Embarked\_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}})[data['Survived']==1]['Embarked'].value\_counts()

Embarked\_not\_survived = data.replace({'Survived': {1: 'Survived', 0: 'Not-survived'}})[data['Survived']==0]['Embarked'].value\_counts()

Embarked\_not\_survived = Embarked\_not\_survived.reindex(index = Embarked\_survived.index)

*# make the bar plot*

p5 = axes[1, 0].bar(Embarked\_survived.index, Embarked\_survived.values)

p6 = axes[1, 0].bar(Embarked\_not\_survived.index, Embarked\_not\_survived.values, bottom=Embarked\_survived.values)

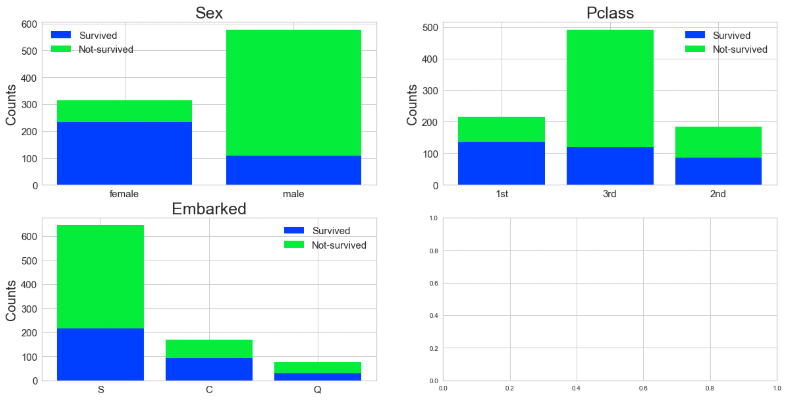
axes[1, 0].set\_title('Embarked', fontsize=25)

axes[1, 0].set\_ylabel('Counts', fontsize=20)

axes[1, 0].tick\_params(axis='both', labelsize=15)

axes[1, 0].legend((p5[0], p6[0]), ('Survived', 'Not-survived'), fontsize = 15)

plt.show()



# Part 2

## Assignment Task

Complete the Titanic Case Study Part 2 tutorial. This will be a complete Analysis Case study but Part 2 is the Feature and Dimensionality Reduction part. I have provided sample code for you to use as you go through the tutorial. I recommend that you comment out the steps and run them separately so you can fully understand what you are doing for each step of the analysis. As you go through each step, take screenshots to “prove” to me that you successfully completed each step. Paste your screenshots into a Word document and submit that Word document to the Assignment submission link.

In [11]:

*#Titanic Tutorial Part 2*

*#Graphics Analysis*

*#Feature Reduction (Extraction/Selection)*

*#Filling in Missing Values*

*#For Part 2 of the Titanic Tutorial, complete Steps 11-13.*

**import** **pandas** **as** **pd**

**import** **yellowbrick**

In [12]:

*# Step 11 - fill in missing values and eliminate features*

*#fill the missing age data with median value*

**def** fill\_na\_median(data, inplace=**True**):

**return** data.fillna(data.median(), inplace=inplace)

fill\_na\_median(data['Age'])

In [13]:

*# check the result*

print(data['Age'].describe())

count 891.000000

mean 29.361582

std 13.019697

min 0.420000

25% 22.000000

50% 28.000000

75% 35.000000

max 80.000000

Name: Age, dtype: float64

In [14]:

*# fill with the most represented value*

**def** fill\_na\_most(data, inplace=**True**):

**return** data.fillna('S', inplace=inplace)

fill\_na\_most(data['Embarked'])

In [15]:

*# check the result*

print(data['Embarked'].describe())

count 891

unique 3

top S

freq 646

Name: Embarked, dtype: object

In [16]:

*# import package*

**import** **numpy** **as** **np**

*# log-transformation*

**def** log\_transformation(data):

**return** data.apply(np.log1p)

data['Fare\_log1p'] = log\_transformation(data['Fare'])

In [17]:

*# check the data*

print(data.describe())

PassengerId Survived Pclass Age SibSp \

count 891.000000 891.000000 891.000000 891.000000 891.000000

mean 446.000000 0.383838 2.308642 29.361582 0.523008

std 257.353842 0.486592 0.836071 13.019697 1.102743

min 1.000000 0.000000 1.000000 0.420000 0.000000

25% 223.500000 0.000000 2.000000 22.000000 0.000000

50% 446.000000 0.000000 3.000000 28.000000 0.000000

75% 668.500000 1.000000 3.000000 35.000000 1.000000

max 891.000000 1.000000 3.000000 80.000000 8.000000

Parch Fare Fare\_log1p

count 891.000000 891.000000 891.000000

mean 0.381594 32.204208 2.962246

std 0.806057 49.693429 0.969048

min 0.000000 0.000000 0.000000

25% 0.000000 7.910400 2.187218

50% 0.000000 14.454200 2.737881

75% 0.000000 31.000000 3.465736

max 6.000000 512.329200 6.240917

In [18]:

*# Step 12 - adjust skewed data (fare)*

*# check the distribution using histogram*

*# set up the figure size*

*#%matplotlib inline*

plt.rcParams['figure.figsize'] = (10, 5)

plt.hist(data['Fare\_log1p'], bins=40)

plt.xlabel('Fare\_log1p', fontsize=20)

plt.ylabel('Counts', fontsize=20)

plt.tick\_params(axis='both', labelsize=15)

plt.show()

In [19]:

*# Step 13 - convert categorical data to numbers*

*# get the categorical data*

cat\_features = ['Pclass', 'Sex', "Embarked"]

data\_cat = data[cat\_features]

data\_cat = data\_cat.replace({'Pclass': {1: '1st', 2: '2nd', 3: '3rd'}})

*# One Hot Encoding*

data\_cat\_dummies = pd.get\_dummies(data\_cat)

*# check the data*

print(data\_cat\_dummies.head(8))

Pclass\_1st Pclass\_2nd Pclass\_3rd Sex\_female Sex\_male Embarked\_C \

0 0 0 1 0 1 0

1 1 0 0 1 0 1

2 0 0 1 1 0 0

3 1 0 0 1 0 0

4 0 0 1 0 1 0

5 0 0 1 0 1 0

6 1 0 0 0 1 0

7 0 0 1 0 1 0

Embarked\_Q Embarked\_S

0 0 1

1 0 0

2 0 1

3 0 1

4 0 1

5 1 0

6 0 1

7 0 1

# Part 3

## Assignment Tasks

## Complete the Titanic Case Study Part 3 tutorial. Now this is a complete Analysis Case study! Part 3 is Model Evaluation and Selection. I have provided sample code for you to use as you go through the tutorial. I recommend that you comment out the steps and run them separately so you can fully understand what you are doing for each step of the analysis. As you go through each step, take screenshots to “prove” to me that you successfully completed each step. Paste your screenshots into a Word document and submit that Word document to the Assignment submission link.

In [20]:

*#Titanic Tutorial Part 3*

*#Graphics Analysis*

*#Feature Reduction (Extraction/Selection)*

*#Filling in Missing Values*

*#Split\_Train\_Test*

*#Model Selection and Evaluation*

*#For Part 3 of the Titanic Tutorial, complete Steps 14-15.*

In [21]:

*#Step 14 - create a whole features dataset that can be used for train and validation data splitting*

*# here we will combine the numerical features and the dummie features together*

features\_model = ['Age', 'SibSp', 'Parch', 'Fare\_log1p']

data\_model\_X = pd.concat([data[features\_model], data\_cat\_dummies], axis=1)

*# create a whole target dataset that can be used for train and validation data splitting*

data\_model\_y = data.replace({'Survived': {1: 'Survived', 0: 'Not\_survived'}})['Survived']

*# separate data into training and validation and check the details of the datasets*

*# import packages*

**from** **sklearn.model\_selection** **import** train\_test\_split

*# split the data*

X\_train, X\_val, y\_train, y\_val = train\_test\_split(data\_model\_X, data\_model\_y, test\_size =0.3, random\_state=11)

*# number of samples in each set*

print("No. of samples in training set: ", X\_train.shape[0])

print("No. of samples in validation set:", X\_val.shape[0])

*# Survived and not-survived*

print('**\n**')

print('No. of survived and not-survived in the training set:')

print(y\_train.value\_counts())

print('**\n**')

print('No. of survived and not-survived in the validation set:')

print(y\_val.value\_counts())

No. of samples in training set: 623

No. of samples in validation set: 268

No. of survived and not-survived in the training set:

Not\_survived 373

Survived 250

Name: Survived, dtype: int64

No. of survived and not-survived in the validation set:

Not\_survived 176

Survived 92

Name: Survived, dtype: int64

In [22]:

*# Step 15 - Eval Metrics*

**from** **sklearn.linear\_model** **import** LogisticRegression

**from** **yellowbrick.classifier** **import** ConfusionMatrix

**from** **yellowbrick.classifier** **import** ClassificationReport

**from** **yellowbrick.classifier** **import** ROCAUC

*# Instantiate the classification model*

model = LogisticRegression()

*#The ConfusionMatrix visualizer taxes a model*

classes = ['Not\_survived','Survived']

cm = ConfusionMatrix(model, classes=classes, percent=**False**)

*#Fit fits the passed model. This is unnecessary if you pass the visualizer a pre-fitted model*

cm.fit(X\_train, y\_train)

*#To create the ConfusionMatrix, we need some test data. Score runs predict() on the data*

*#and then creates the confusion\_matrix from scikit learn.*

cm.score(X\_val, y\_val)

*# change fontsize of the labels in the figure*

**for** label **in** cm.ax.texts:

label.set\_size(20)

*#How did we do?*

cm.poof()

*# Precision, Recall, and F1 Score*

*# set the size of the figure and the font size*

*#%matplotlib inline*

plt.rcParams['figure.figsize'] = (15, 7)

plt.rcParams['font.size'] = 20

*# Instantiate the visualizer*

visualizer = ClassificationReport(model, classes=classes)

visualizer.fit(X\_train, y\_train) *# Fit the training data to the visualizer*

visualizer.score(X\_val, y\_val) *# Evaluate the model on the test data*

g = visualizer.poof()

*# ROC and AUC*

*#Instantiate the visualizer*

visualizer = ROCAUC(model)

visualizer.fit(X\_train, y\_train) *# Fit the training data to the visualizer*

visualizer.score(X\_val, y\_val) *# Evaluate the model on the test data*

g = visualizer.poof()

