DSC 550

Week 10 – Original Cast Study

Complete Code & Screen Shots

import pandas as pd

import yellowbrick

#Step 1: Load the data from the “games.csv” file into a DataFrame

data = pd.read\_csv("C:/Users/Dasun/Desktop/Bellevue/DSC\_550/games.csv")

# Step 2: Display the dimensions of the file

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

#Step 3: Display the first 5 rows of data so you can see the column headings and the type of data for each column

data.head(5)

Finding missing values

data.isnull().sum()

Finding column types

data.dtypes

#Step 4: Converting rating columns to integer types

data.astype({'white\_rating': 'int32'}).dtypes

data.astype({'black\_rating': 'int32'}).dtypes

#Step 5: Looking at summary information

data.describe()

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

#Step 6: Creating Histograms using only the first 1000 rows

data = data[:1000]

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 = ['opening\_eco', 'white\_rating', 'black\_rating', 'victory\_status']

xaxes = num\_features

yaxes = ['Count', 'Count', 'Count', 'Count']

# 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()

#Step 7: Creating Bar Charts

# set up the figure size

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\_Winner = data.groupby('winner').size().reset\_index(name='Count')['winner']

Y\_Winner = data.groupby('winner').size().reset\_index(name='Count')['Count']

# make the bar plot

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

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

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

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

# make the data read to feed into the visulizer

X\_victory\_status = data.groupby('victory\_status').size().reset\_index(name='Count')['victory\_status']

Y\_victory\_status = data.groupby('victory\_status').size().reset\_index(name='Count')['Count']

# make the bar plot

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

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

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

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

# make the data read to feed into the visulizer

X\_rated = data.groupby('rated').size().reset\_index(name='Count')['rated']

Y\_rated = data.groupby('rated').size().reset\_index(name='Count')['Count']

# make the bar plot

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

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

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

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

# make the data read to feed into the visulizer

X\_increment\_code = data.groupby('increment\_code').size().reset\_index(name='Count')['increment\_code']

Y\_increment\_code = data.groupby('increment\_code').size().reset\_index(name='Count')['Count']

# make the bar plot

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

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

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

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

plt.show()

#Step 8: Finding the Pearson Ranking

#set up the figure size

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

num\_features = ['turns', 'white\_rating', 'black\_rating']

X = data[num\_features].as\_matrix()

# 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="C:/Users/Dasun/Desktop/Bellevue/DSC\_550/case\_study\_1.png") # Draw/show/poof the data

plt.show()

# Step 9: Using Parallel Coordinates visualization for comparison

#set up the figure size

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 = ['White', 'Black', 'Draw']

# 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].as\_matrix()

y = data.winner.as\_matrix()

# 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="C:/Users/Dasun/Desktop/Bellevue/DSC\_550/case\_study\_2.png") # Draw/show/poof the data

plt.show();

# Step 10: Using Stack Bar Charts to compare the winner and victory status based on the game being rated or not

#set up the figure size

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

# make subplots

fig, axes = plt.subplots(nrows = 1, ncols = 2)

# make the data read to feed into the visulizer

winner\_rated = data[data['rated']== 1]['winner'].value\_counts()

winner\_unrated = data[data['rated']!= 1]['winner'].value\_counts()

winner\_unrated = winner\_unrated.reindex(index = winner\_rated.index)

print(len(rated))

# make the bar plot

p1 = axes[0].bar(winner\_rated.index, winner\_rated.values)

p2 = axes[0].bar(winner\_unrated.index, winner\_unrated.values, bottom = winner\_rated.values)

print(p1)

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

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

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

axes[0].legend((p1[0], p2[0]), ('rated', 'unrated'), fontsize = 15)

# make the data read to feed into the visulizer

victory\_rated = data[data['rated']== 1]['victory\_status'].value\_counts()

victory\_unrated = data[data['rated']!= 1]['victory\_status'].value\_counts()

unrated = unrated.reindex(index = victory\_rated.index)

# make the bar plot

p3 = axes[1].bar(victory\_rated.index, victory\_rated.values)

p4 = axes[1].bar(victory\_unrated.index, victory\_unrated.values, bottom = victory\_rated.values)

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

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

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

axes[1].legend((p3[0], p4[0]), ('rated', 'unrated'), fontsize = 15)

plt.show()

# Step 11: Using PCA to identify feature importance

data2 = data[['rated', 'turns', 'white\_rating', 'black\_rating']]

import numpy

from sklearn.decomposition import PCA

array = data2.values

X = array[:,0:3]

Y = array[:,3]

# feature extraction

pca = PCA(n\_components=3)

fit = pca.fit(X)

# summarize components

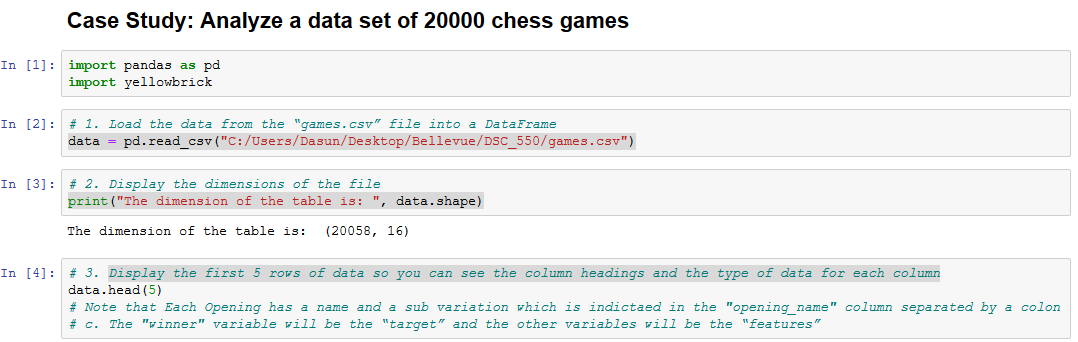
print("Explained Variance: %s" % fit.explained\_variance\_ratio\_)

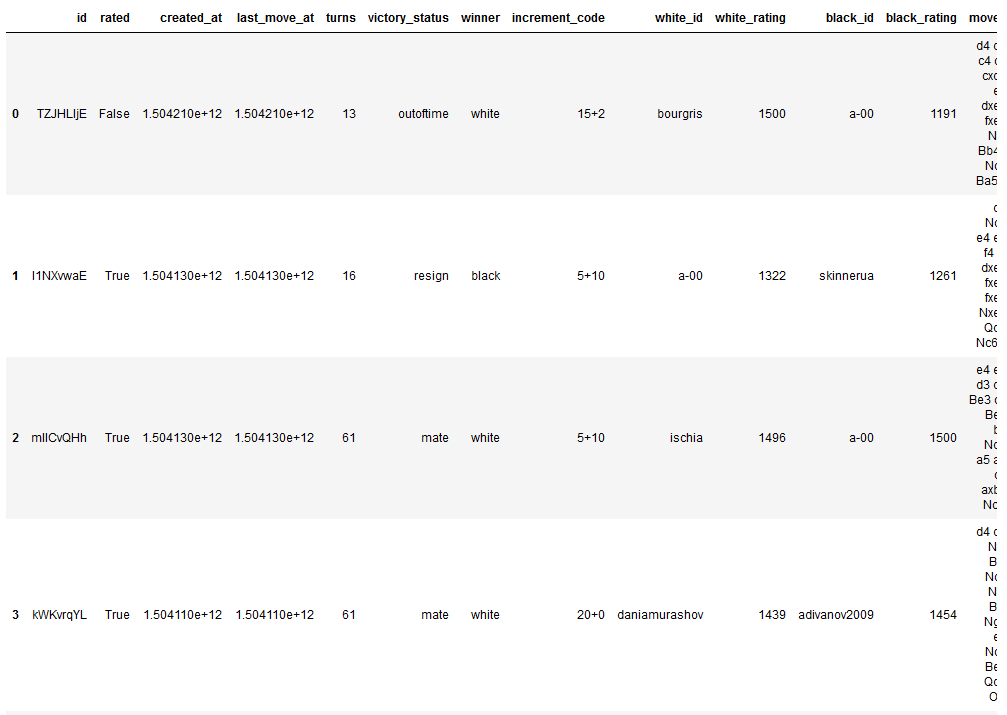
print(fit.components\_)

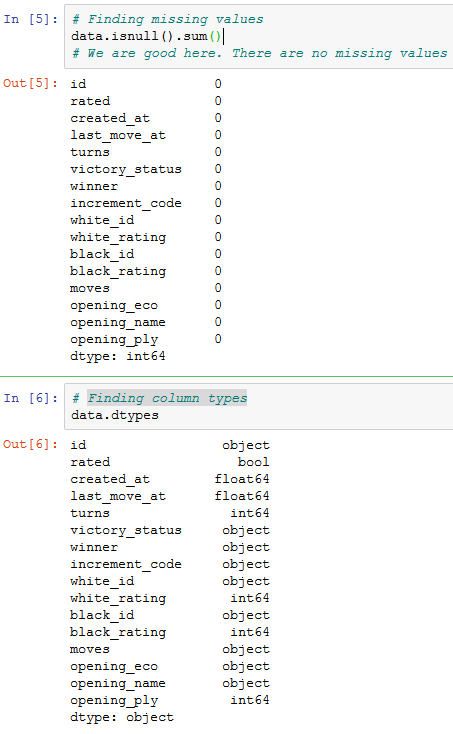
# Step 12: Checking the variance for each column

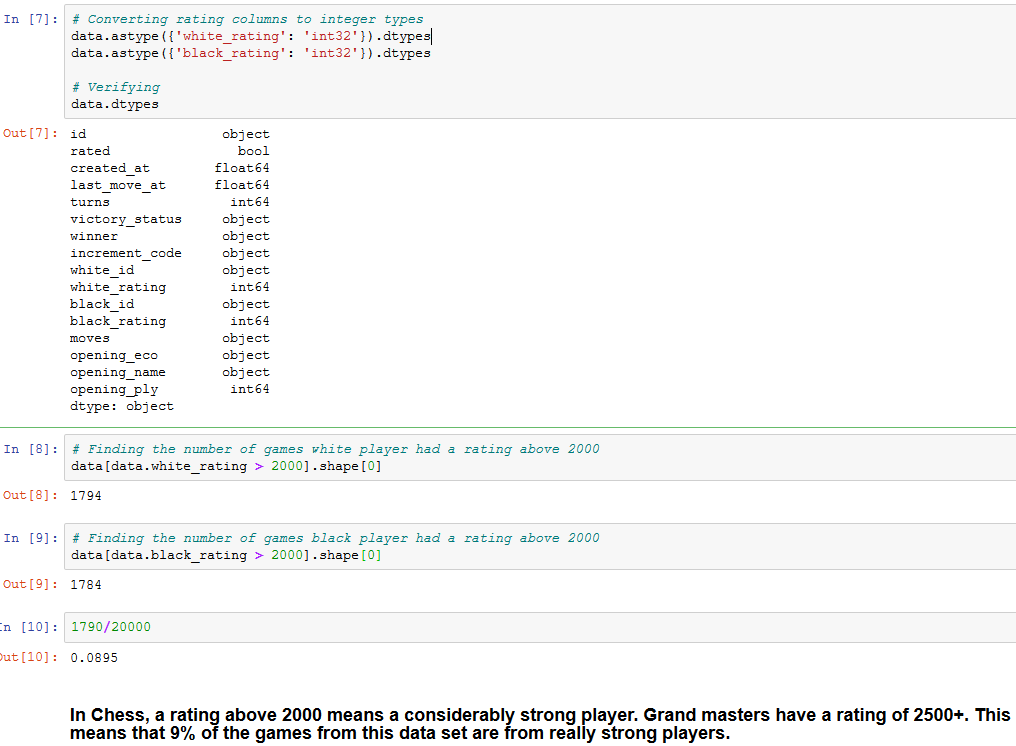
data.var(axis=0)

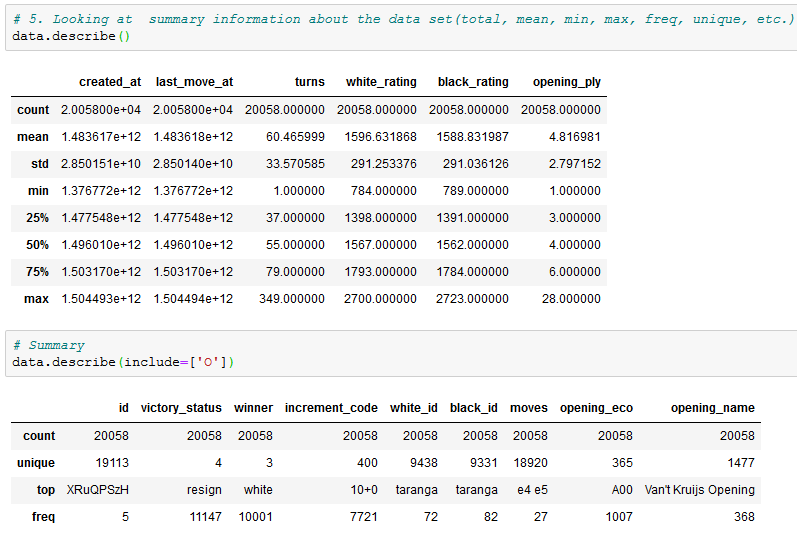
Screenshots

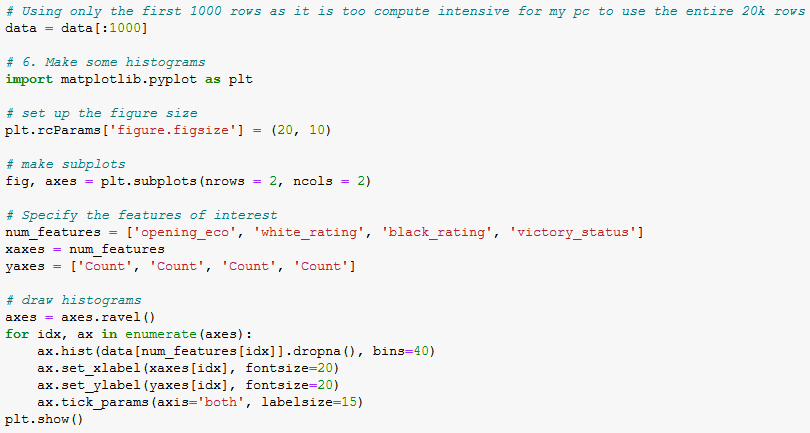


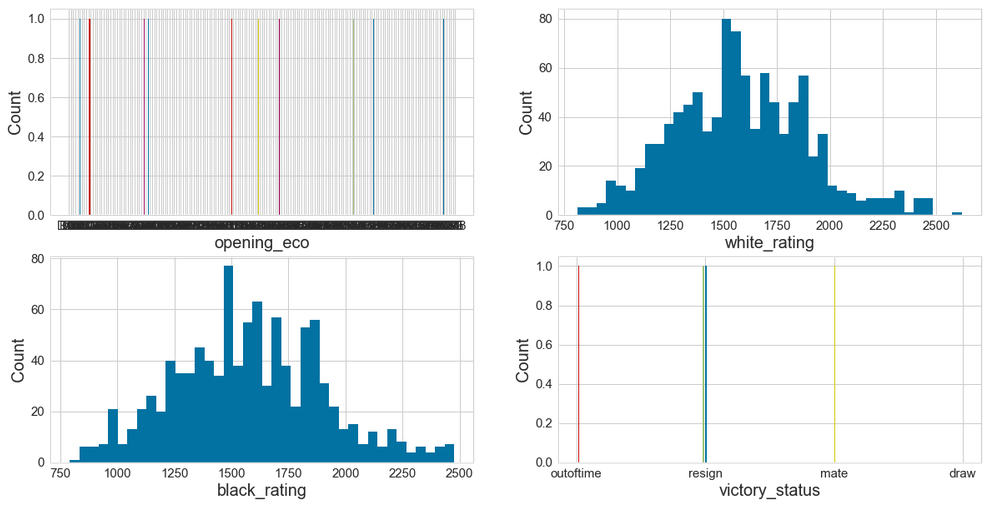


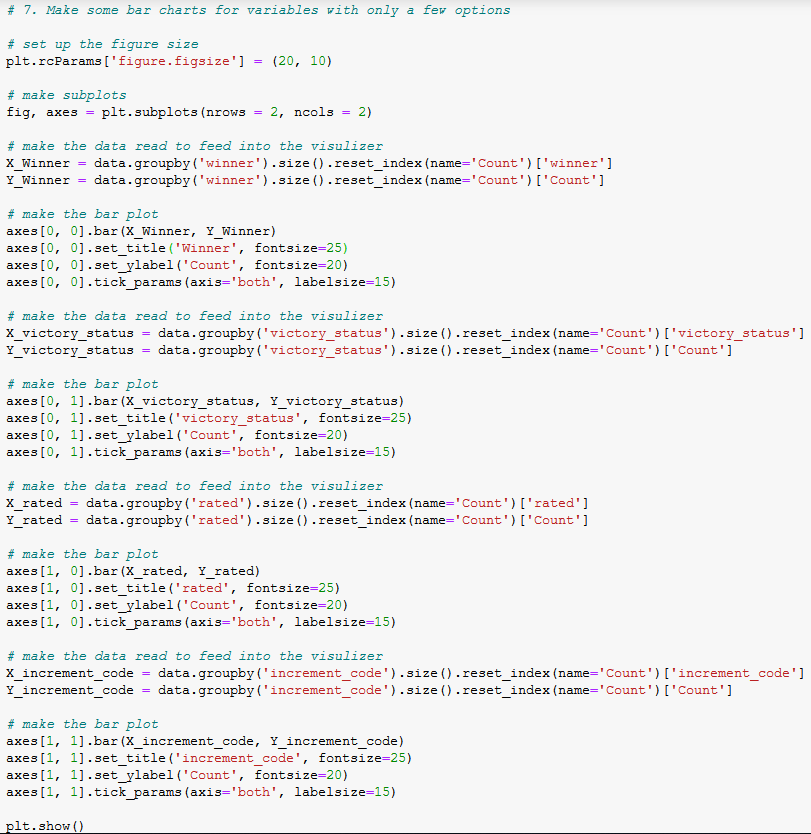


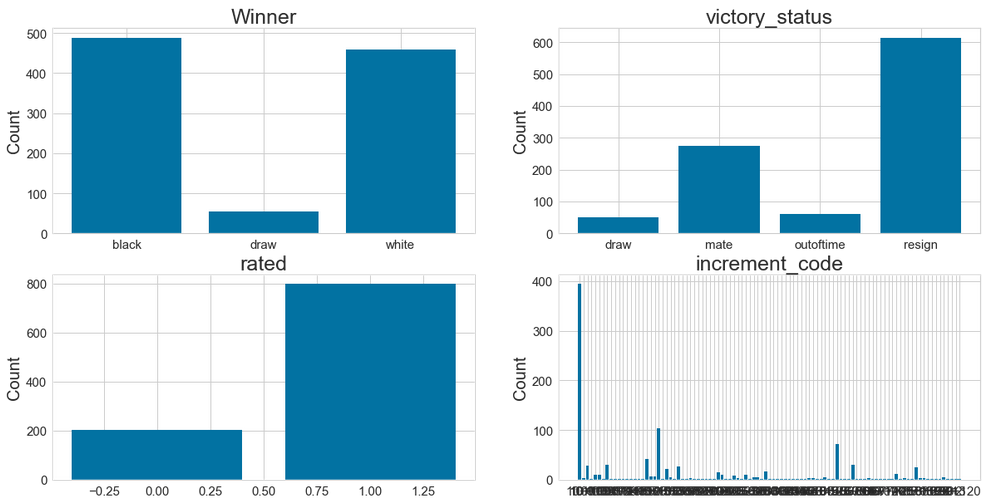


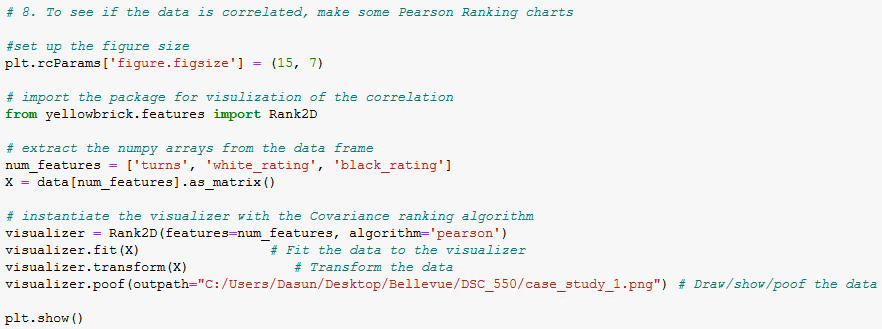


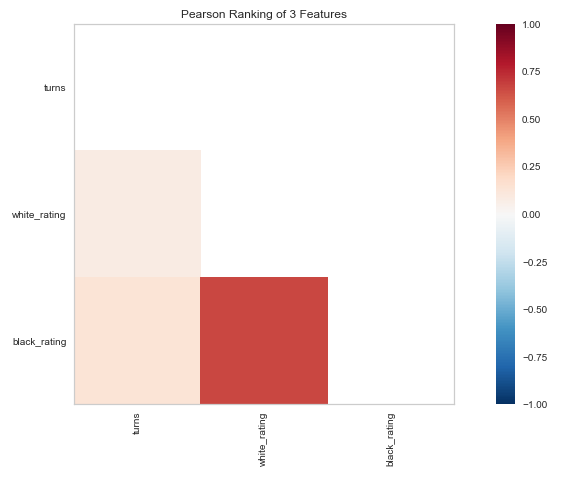




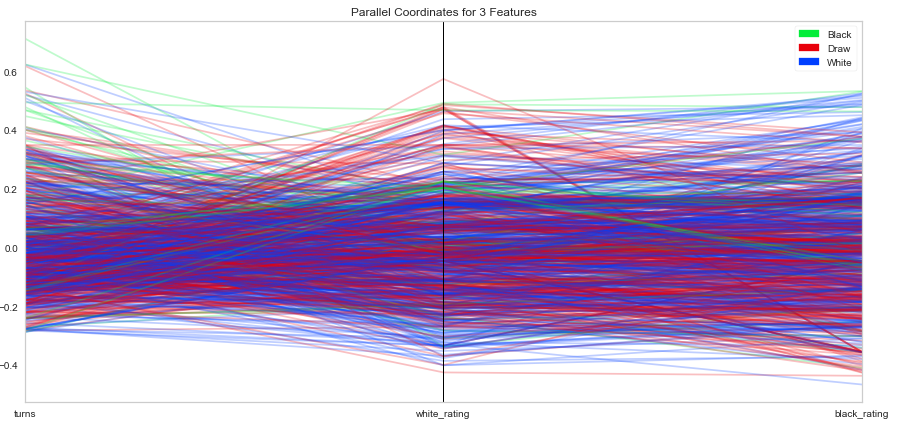


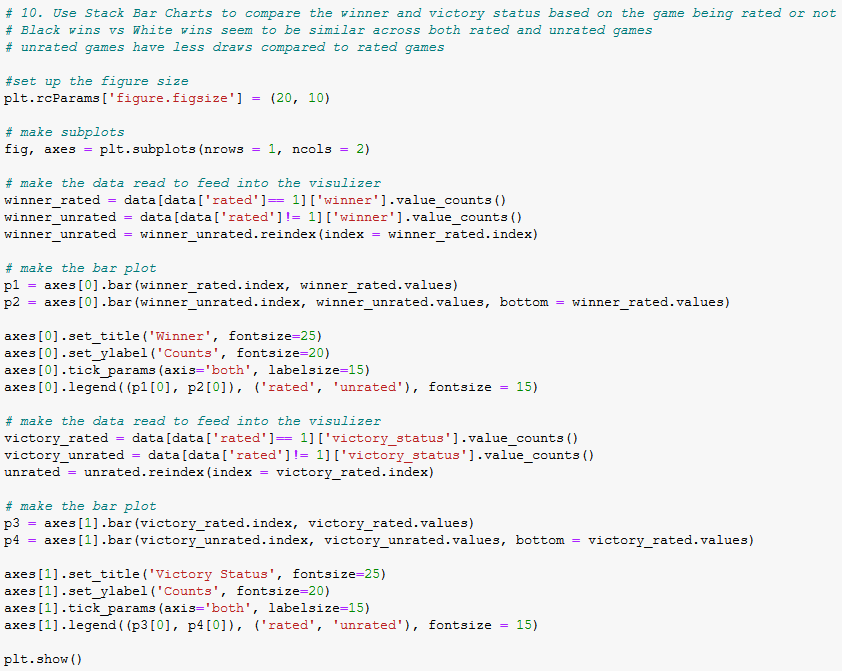


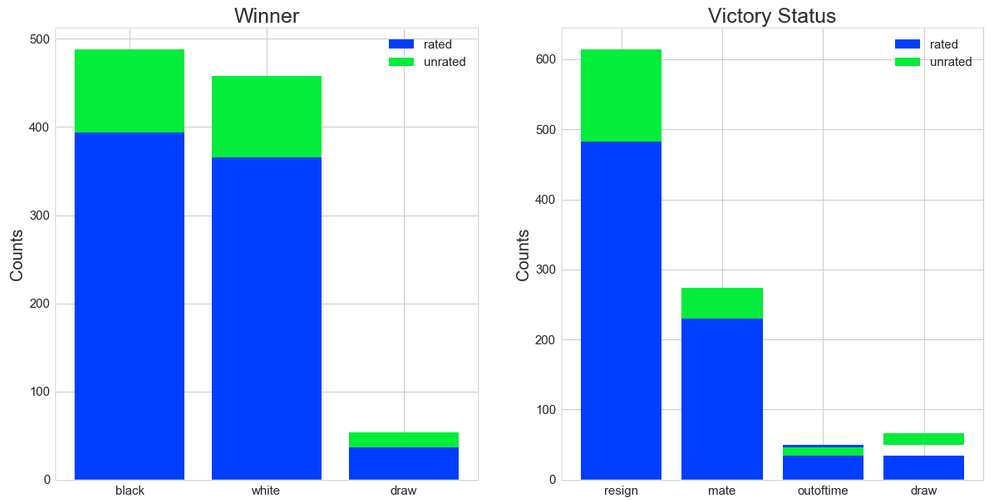


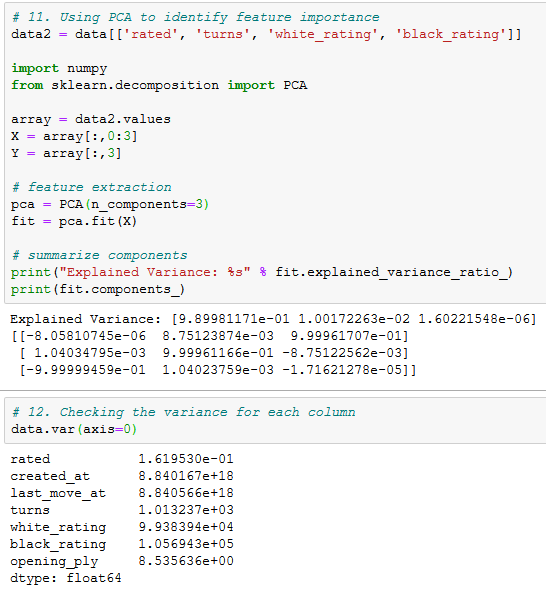












A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated