PCA_NBA

December 10, 2021

1 Principal Component Analysis on CBA/NBA Stats

Author: Eric Luong

Abstract: In order to determine if there is a correlation between player CBA Stats and NBA Performance, I used Principal Component Analysis on player CBA stats and analyzed the correlation between the principal components and actual NBA player data. Using a skree plot to visualize the variance, we see that ~80% of the variance is captured by the first two principal components. Since there seemed to be a nonlinear correlation, I decided to use both spearman and pearson r values to analyze the correlation. The correlation between the first principal component and NBA points per game showed a spearman-r value of 0.30574484244029815 and pearson-r value of 0.321405633114988, while the p values were 4.288084864102229e-16 and 1.0395014143835297e-17. Although the correlation seems a bit low, there are a large amount of data points. Since there might be further bias influencing the correlation, I decided to apply Bonferoni's p-value adjustment. After performing a Bonferoni p-value adjustment, we see that the p(4.288084864102229e-16, 7.396449704142012e-05). Since the p-value is still significantly less than the threshold after adjusting the threshold, we can reject the null hypothesis and show that there is a correlation between CBA performance and NBA performance.

Introduction: Scouts and trainers are constantly looking over spreadsheets and numbers to figure out which basketball players are most likely to succeed, starting with their college stats. In order to see if there is a way to predict future NBA performance with CBA stats, I used college player's statistics and box score information from the last 7 years. Since not all the data is useful, I did some data cleaning to keep the college stats that seemed relevant to performance and the NBA stats for points per game. Since not all of the features seemed relevant, I applied principal component analysis to the college player statistics.

[1]:
```
# import packages
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

1.0.1 Importing the data

[2]:
```
college = pd.read_csv('final_proj_data/basketball/college.csv')
teamBoxScore = pd.read_csv('final_proj_data/basketball/teamBoxScore.csv')
standings = pd.read_csv('final_proj_data/basketball/standings.csv')
playerBoxScore = pd.read_csv('final_proj_data/basketball/playerBoxScore.csv')
officialBoxScore = pd.read_csv('final_proj_data/basketball/officialBoxScore.
↪csv')
```

The Data: The two data sets I will be using are the college stats and player Box Score stats. Since not all the features are necessary in this analysis, features such a active data, birth date, etc., I created new data frames with the cleaned up name column along with performance stats such as points per game, field goal percentage, etc.

1.0.2 Data Cleaning and Data Exploration

Cleaning CBA Data:

[3]:
```
# selecting columns of college stats

collegeTwo = college[['name', 'NCAA_ppg', 'NCAA__3ptpg', 'NCAA_fgapg',␣
↪'NCAA_fgpct', 'NCAA_fgpg',
                      'NCAA_ft', 'NCAA_ftapg', 'NCAA_ftpg']]
```

[4]:
```
# copy original data for cleaning

collegeTwoCleaned = pd.DataFrame(collegeTwo)


for column in collegeTwoCleaned.columns[1:]:
    collegeTwoCleaned[column] = collegeTwo.groupby('name')[column].
↪transform(lambda x: x.fillna(x.mean()))

collegeTwoCleaned = collegeTwoCleaned.groupby('name').mean()
```

Cleaning NBA Data:

[5]:
```
# Creating data frame with the name and points for each game

playerBoxScoreTwo = playerBoxScore[['playFNm', 'playLNm', 'playPTS']]
playerBoxScoreTwo['playNm'] = playerBoxScoreTwo['playFNm'] + ' ' +␣
↪playerBoxScore['playLNm']
playerBoxScoreTwo = playerBoxScoreTwo.drop(['playFNm', 'playLNm'], axis = 1)
column_titles = ['playNm', 'playPTS']
playerBoxScoreTwo = playerBoxScoreTwo.reindex(columns = column_titles)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

[6]:
```
# groups by name and computes avg points per game

playerBoxScoreTwo = playerBoxScoreTwo.groupby('playNm').mean()

playerBoxScoreTwo.index.names = ['name']

playerBoxScoreTwo.head(5)
```

[6]:

name	playPTS
A.J. Price	5.567568
Aaron Brooks	7.675603
Aaron Gordon	11.418251
Aaron Gray	2.329114
Aaron Hammons	2.181818

[7]:
```
joinedDf = collegeTwoCleaned.join(playerBoxScoreTwo)
```

[8]:
```
cleaned_data = joinedDf.drop('playPTS', axis = 1).dropna() # dropped playPTS␣
↪for PCA
```

1.0.3 Method and Experiments:

[9]:
```
# centering the data
centered_data = cleaned_data - cleaned_data.mean()
centered_data = (centered_data / centered_data.std()).dropna()
```

[10]:
```
# rejoin the 'playPTS' column and drops na rows
centered_data_joined = centered_data.join(playerBoxScoreTwo).dropna()
```

[11]:
```
centered_data = centered_data_joined[centered_data.columns]
```

[12]:
```
# computes rank k approximation of data
def compute_rank_k_approximation(data, k):
    u, s, vt = np.linalg.svd(data, full_matrices = False)
    return pd.DataFrame(u[:, 0:k] @ np.diag(s[0:k]) @ vt[0:k, :], columns =␣
↪data.columns)
```

```
joined_data_rank_3_approximation = compute_rank_k_approximation(centered_data,␣
↪3)

joined_data_rank_3_approximation.head(5)
```

[12]:

	NCAA_ppg	NCAA__3ptpg	NCAA_fgapg	NCAA_fgpct	NCAA_fgpg	NCAA_ft	\
0	0.055898	0.832449	0.233246	-0.905525	-0.081546	0.669556	
1	-0.004414	1.085195	0.182353	-1.356800	-0.287711	1.148808	
2	0.047345	-0.751693	0.106542	1.462646	0.603681	-1.744456	
3	-0.778923	-1.171595	-0.874943	1.114689	-0.504569	-1.137366	
4	-0.111064	0.859072	0.027258	-1.184267	-0.383906	1.040002	

	NCAA_ftapg	NCAA_ftpg
0	-0.401546	-0.185952
1	-0.413733	-0.054990
2	-0.335704	-0.836830
3	-0.347966	-0.657025
4	-0.354197	-0.028811

From the approximation, we can see that even a rank 3 approximation gives relatively reasonable estimates.

[13]:
```
# helper function returns s from PCA
def return_s(data, k):
    u, s, vt = np.linalg.svd(data, full_matrices = False)
    return s
```

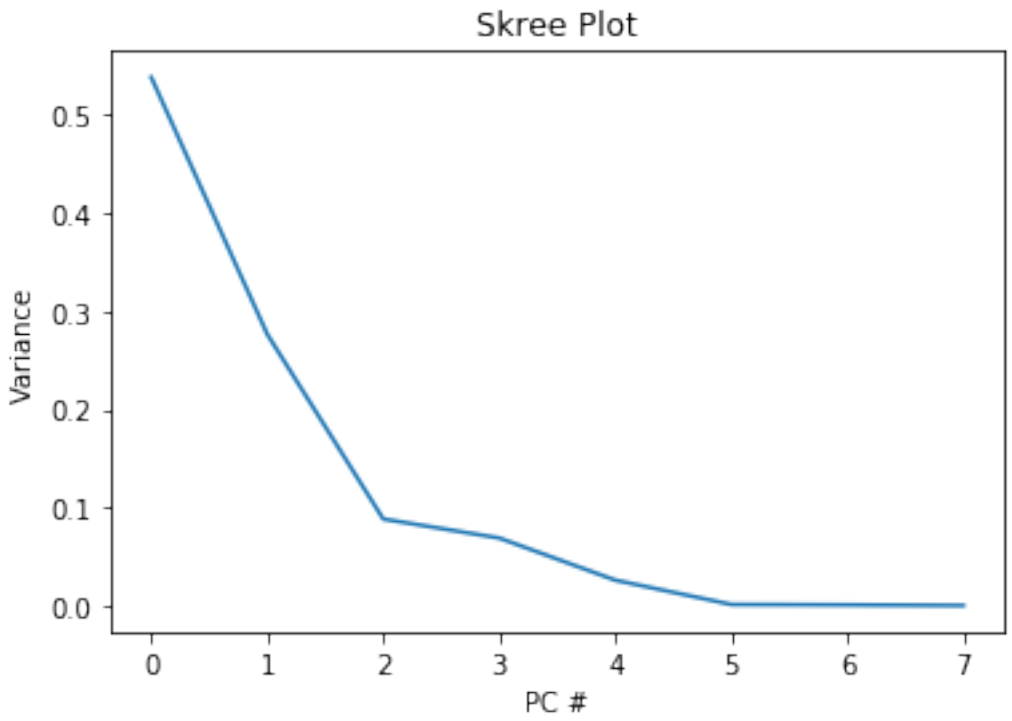
[14]:
```
# helper function returns vt from PCA
def return_vt(data, k):
    u, s, vt = np.linalg.svd(data, full_matrices = False)
    return vt

vt = return_vt(centered_data, 3)
```

Skree Plot and Variance

[15]:
```
s = return_s(centered_data, 3)
plt.plot(s ** 2 / sum(s ** 2))
plt.title("Skree Plot")
plt.xlabel('PC #')
plt.ylabel('Variance')
```

[15]: Text(0, 0.5, 'Variance')

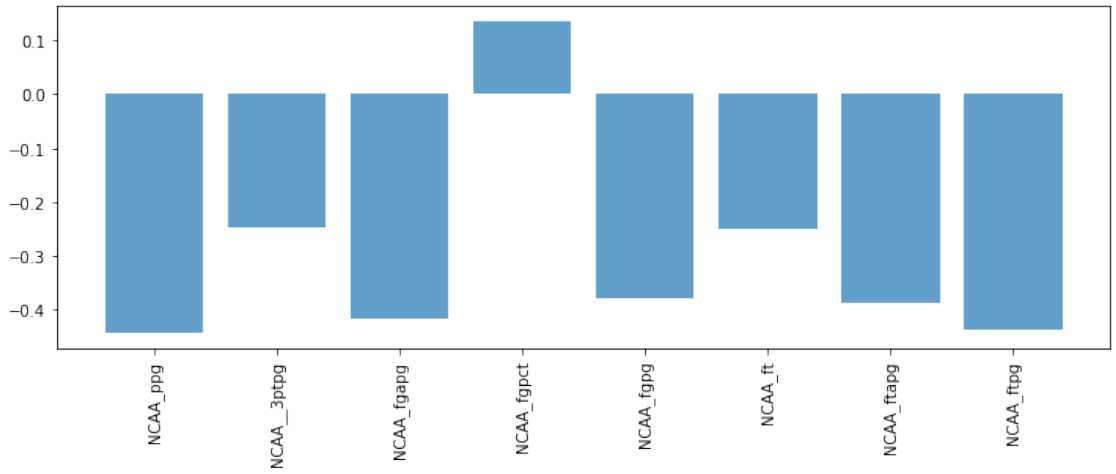


From this Skree Plot, we see that ~80% of the variance is captured by the first two principal components. However, since the third pc also captures close to the second, we will use the first three principal components in our data visualization.

[16]:
```
def plot_pc(col_names, vt, k):
    plt.bar(col_names, vt[k, :], alpha=0.7)
    plt.xticks(col_names, rotation=90);

col_names = centered_data.columns

with plt.rc_context({"figure.figsize": (12, 4)}):
    plot_pc(col_names, vt, 0);
```



1.0.4 PCA Visualizations

[17]:
```
from sklearn import decomposition
```

[18]:
```
# performs pca and keeps first 3 principal components

pca = decomposition.PCA(n_components = 3)
pca_sk = pd.DataFrame(pca.fit_transform(centered_data))
```

[19]:
```
# reset index for plotting
centered_data_joined = centered_data_joined.reset_index()
```

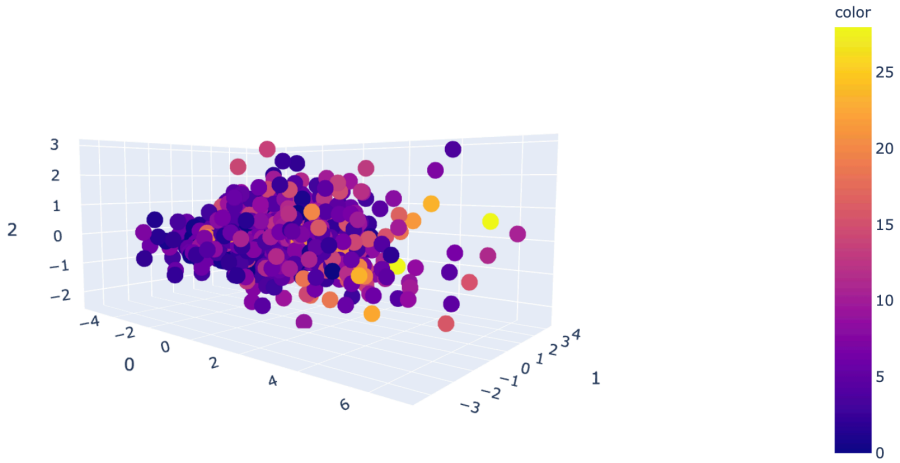
[44]:
```
import plotly.express as px
import plotly

fig = px.scatter_3d(pca_sk, x = 0, y = 1, z = 2, color =␣
↪centered_data_joined['playPTS'], hover_name = centered_data_joined['name'])
#fig.show()

fig = px.scatter(pca_sk, x = 0, y = 1, color = centered_data_joined['playPTS'],␣
↪hover_name = centered_data_joined['name'])
#fig.show()
```

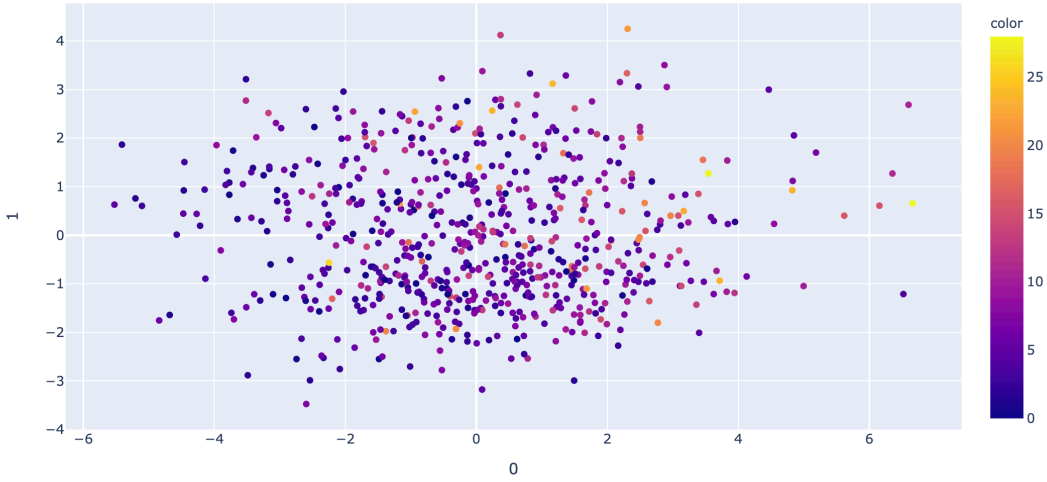
[41]:
```
# import png versions of above visualizations
from IPython.display import Image
Image("PCA_3D.png") # 3D scatterplot of PC0, 1, 2, and NBA_ppg
```

[41]:



[43]:
```
Image("PCA_2D.png") # 2D scatterplot of PC0, 1 and NBA_ppg
```

[43]:



1.0.5 Statistics

[21]:
```
import scipy.stats as stats
```

Since the data seems nonlinear, I applied both pearsonr and spearmanr to Principal Component 1 and Avg points per game. Principal component 1 captures

a majority of the variance so it is the best feature to use to compute the correlation.

[22]:
```
# pearson r and p value of Principal component 1 vs Avg points per NBA game
stats.pearsonr(pca_sk[0], centered_data_joined['playPTS'])
```

[22]:
```
(0.32140563311498804, 1.0395014143834707e-17)
```

[23]:
```
# spearman r and p value of Principal component 1 vs Avg points per NBA game
stats.spearmanr(pca_sk[0], centered_data_joined['playPTS'])
```

[23]:
```
SpearmanrResult(correlation=0.30574484244029815, pvalue=4.288084864102229e-16)
```

[24]:
```
# bonferoni p-value adjustment
def bonferoni(df):
    r, p = stats.pearsonr(pca_sk[0], centered_data_joined['playPTS'])
    adjusted_thresh = 0.05 / len(centered_data_joined['playPTS'])
    return adjusted_thresh

print((bonferoni(centered_data_joined), 1.0395014143834707e-17))
```

```
(7.396449704142012e-05, 1.0395014143834707e-17)
```

1.0.6 Analysis and Conclusion

From the 3D scatterplot, the color represents the avg ppg (points per game) in the NBA, with yellow being a high ppg. Along the axis of principal component 0, the more positive values have a higher concentration of yellow and orange points compared to the negative values which contain a large concentration of purple points. This varies more when looking from the axis of pc1 or pc2.

However, when plotting the data in a 2D scatterplot, we can better visualize how the first two principal components can help distinguish player performance in the NBA. At first glance we see more yellow and orange for positive pc0, but not much of a difference along the pc1 axis. This fits what the earlier skree plot displayed, that one feature contributes to over 50% of the variance. But with two PC's plotted instead of 3, we can see how the majority of yellow and orange points are located closer to the positive PC0 and PC1. This goes to show how much of the variance is caught by the first two principal components.

Originally, I tried to do a PCA on another data set with NBA stats joined into the data set. The k-rank approximation through this method also showed that even a 3-rank approximation yielded relatively reasonable points for NBA stats. This just goes to confirm the skree plot we have above, so it would have been redundant to continue. Another method that didn't yield any obvious results was a linear regression on college average points per game vs NBA average points per game. Although I thought it would be a feature that would yield a nice linear regression, it did not work out the way I hoped. Even though this was a challenge

at first, this showed that there was a nonlinear relationship, which led me to try analyzing the correlation between the primary principal component and NBA average points per game.

I think more complete data with less NA values scatterred throughout the data set would help solidify the analysis. Since I had to drop many rows of NA, the data set does not encapsulate all the player stats of the last 7 years. There could be a slight bias in which players were kept or which players had less NA values. Some of the ethical concerns I had when facing this problem was whether to just not perform the same methods on the players with missing stats. By not doing so, it could reinforce human biases of already favoring certain players in record keeping.

While the first principal component captured ~50% of the variance, we see that ~80% of the variance is captured by the first two principal components. Since there seemed to be a nonlinear correlation, I decided to use both spearman and pearson r values to analyze the correlation. The correlation between the first principal component and NBA points per game showed a spearman-r value of 0.30574484244029815 and pearson-r value of 0.321405633114988, while the p values were 4.288084864102229e-16 and 1.039501414383529 7e-17. Although the correlation seems a bit low, there are a large amount of data points. Since there might be further bias influencing the correlation, I decided to apply Bonferoni's p-value adjustment. After performing a Bonferoni p-value adjustment, we see that the p(1.5377239857744522e-20, 7.396449704142012e-05). Since the p-value is still significantly less than the threshold after adjusting the threshold, we can reject the null hypothesis and show that there is a correlation between CBA performance and NBA performance.