392 Final

College Basketball Dataset

> Thomas Moore

> Hari Kumar


```
# importing packages
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
from plotnine import *
from sklearn import *
# Linear Regression Model
from sklearn.linear_model import LinearRegression, Ridge, Lasso
# Logistic Regression Model
from sklearn.linear_model import LogisticRegression
#Z-score variables
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, plot_confusion_matrix
# simple TT split cv
from sklearn.model_selection import train_test_split
# k-fold cv
from sklearn.model_selection import KFold
#LOO cv
from sklearn.model_selection import LeaveOneOut
# cross validation metrics
from sklearn.model_selection import cross_val_score
# cross validation metrics
from sklearn.model_selection import cross_val_predict
# model eval
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
from matplotlib import pyplot as plt
%matplotlib inline

from sklearn import preprocessing
from sklearn import utils
```

Variable Descriptions:

- **RK** (Only in cbb20): The ranking of the team at the end of the regular season according to barttorvik
- **TEAM**: The Division I college basketball school (object)
- **CONF**: The Athletic Conference in which the school participates in (object):

  A10 = Atlantic 10, ACC = Atlantic Coast Conference, AE = America East, Amer = American, ASun = ASUN, B10 = Big Ten, B12 = Big 12, BE = Big East, BSky = Big Sky, BSth = Big South, BW = Big West, CAA = Colonial Athletic Association, CUSA = Conference USA, Horz = Horizon League, Ivy = Ivy League, MAAC = Metro Atlantic Athletic Conference, MAC = Mid-American Conference, MEAC = Mid-Eastern Athletic Conference, MVC = Missouri Valley Conference, MWC = Mountain West, NEC = Northeast Conference, OVC = Ohio Valley Conference, P12 = Pac-12, Pat = Patriot League, SB = Sun Belt, SC = Southern Conference, SEC = South Eastern Conference, Slnd = Southland Conference, Sum = Summit League, SWAC = Southwestern Athletic Conference, WAC = Western Athletic Conference, WCC = West Coast Conference

G: Number of games played (int 64)

W: Number of games won (int 64)

the rest are floats:

ADJOE: Adjusted Offensive Efficiency (An estimate of the offensive efficiency (points scored per 100 possessions) a team would have against the average Division I defense)

ADJDE: Adjusted Defensive Efficiency (An estimate of the defensive efficiency (points allowed per 100 possessions) a team would have against the average Division I offense)

BARTHAG: Power Rating (Chance of beating an average Division I team)

EFG_O: Effective Field Goal Percentage Shot

EFG_D: Effective Field Goal Percentage Allowed

TOR: Turnover Percentage Allowed (Turnover Rate)

TORD: Turnover Percentage Committed (Steal Rate)

ORB: Offensive Rebound Rate

DRB: Offensive Rebound Rate Allowed

FTR: Free Throw Rate (How often the given team shoots Free Throws)

FTRD: Free Throw Rate Allowed

2P_O: Two-Point Shooting Percentage

2P_D: Two-Point Shooting Percentage Allowed

3P_O: Three-Point Shooting Percentage

3P_D: Three-Point Shooting Percentage Allowed

ADJ_T: Adjusted Tempo (An estimate of the tempo (possessions per 40 minutes) a team would have against the team that wants to play at an average Division I tempo)

WAB: Wins Above Bubble (The bubble refers to the cut off between making the NCAA March Madness Tournament and not making it)

POSTSEASON: Round where the given team was eliminated or where their season ended (R68 = First Four, R64 = Round of 64, R32 = Round of 32, S16 = Sweet Sixteen, E8 = Elite Eight, F4 = Final Four, 2ND = Runner-up, Champion = Winner of the NCAA March Madness Tournament for that given year)

SEED: Seed in the NCAA March Madness Tournament

YEAR: Season

```
# reading in all college basketball data
all_data = pd.read_csv("https://raw.githubusercontent.com/tmoore-byte/392Final/main/cbb.csv")
```

```
# data for just the year 2021
data = pd.read_csv("https://raw.githubusercontent.com/tmoore-byte/392Final/main/cbb21.csv")
data.head()
```

	TEAM	CONF	G	W	ADJOE	ADJDE	BARTHAG	EFG_O	EFG_D	TOR	...	DRB	FT
0	Michigan	B10	24	20	118.1	91.1	0.9521	54.9	44.9	16.3	...	24.8	28.
1	Baylor	B12	24	22	123.2	94.5	0.9548	57.5	49.1	17.6	...	30.9	27.
2	Illinois	B10	29	23	117.7	90.4	0.9539	55.6	46.6	18.2	...	22.2	39.
3	Gonzaga	WCC	26	26	125.4	89.8	0.9791	61.0	47.5	16.1	...	23.4	36.
4	Iowa	B10	29	21	123.5	95.7	0.9491	54.6	48.3	13.3	...	28.6	32.

5 rows × 22 columns

```
# 347 rows, 22 columns
# 347 teams, 22 variables
data.shape
```

```
(347, 22)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 347 entries, 0 to 346
Data columns (total 22 columns):
```

```
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   TEAM     347 non-null    object
 1   CONF     347 non-null    object
 2   G        347 non-null    int64
 3   W        347 non-null    int64
 4   ADJOE    347 non-null    float64
 5   ADJDE    347 non-null    float64
 6   BARTHAG  347 non-null    float64
 7   EFG_O    347 non-null    float64
 8   EFG_D    347 non-null    float64
 9   TOR      347 non-null    float64
 10  TORD     347 non-null    float64
 11  ORB      347 non-null    float64
 12  DRB      347 non-null    float64
 13  FTR      347 non-null    float64
 14  FTRD     347 non-null    float64
 15  2P_O     347 non-null    float64
 16  2P_D     347 non-null    float64
 17  3P_O     347 non-null    float64
 18  3P_D     347 non-null    float64
 19  ADJ_T    347 non-null    float64
 20  WAB      347 non-null    float64
 21  SEED     347 non-null    int64
dtypes: float64(17), int64(3), object(2)
memory usage: 59.8+ KB
```

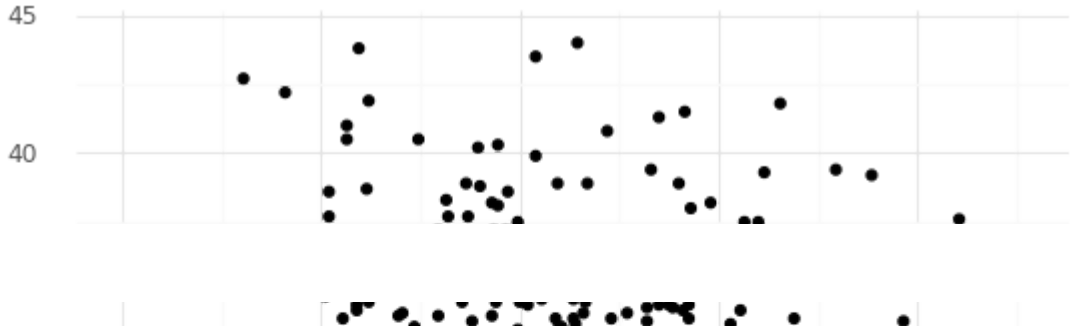
```
# checking for null values
data.isnull().sum(axis=0)

# there are 279 teams that did not make the tournament, SEED = NULL
```

```
TEAM       0
CONF       0
G          0
W          0
ADJOE      0
ADJDE      0
BARTHAG    0
EFG_O      0
EFG_D      0
```

```
TOR        0
TORD       0
ORB        0
DRB        0
FTR        0
FTRD       0
2P_O       0
2P_D       0
3P_O       0
3P_D       0
ADJ_T      0
WAB        0
SEED       0
dtype: int64
```

#

```
# data visualization
(ggplot(data, aes(x = "ADJOE", y = "FTR"))+geom_point()+ theme_minimal())
```



QUESTION 1:

- based on offensive variables 2-point percentage (2P_O), 3 point percentage (3P_O), and free throw rate (FTR):

Which predictors will have the most impact on offensive efficiency (ADJOE)?


```
(ggplot(data, aes("ADJOE"))+ geom_histogram()+ theme_minimal())
```


```
# splitting data into train and test. 75:25
predictors = ["2P_O", "3P_O", "FTR"]
X = data[predictors]
y = data["ADJOE"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```


```
# zscoring
z = StandardScaler()
X_train[predictors] = z.fit_transform(X_train[predictors])
X_test[predictors] = z.fit_transform(X_test[predictors])

# building and fitting model to specified 2021 data
lr = LinearRegression()
lr.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

ADJOE

```
print("Train: " , mean_absolute_error(y_train, lr.predict(X_train)))
print("Test: " , mean_absolute_error(y_test, lr.predict(X_test)))
```

```
Train:  4.2601399500160175
Test:  4.413265940821736
```

```
lr_rr = Ridge()

lr_rr.fit(X_train, y_train)

print("Train: " , mean_absolute_error(y_train, lr_rr.predict(X_train)))
print("Test: " , mean_absolute_error(y_test, lr_rr.predict(X_test)))
```

```
Train:  4.259415999696598
Test:  4.41253756291195
```

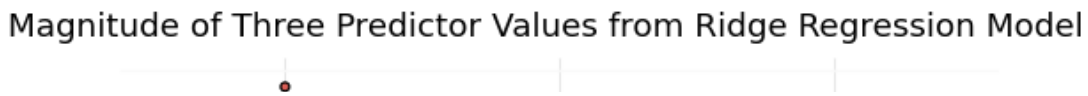
```
lr_rrDf = pd.DataFrame({"Coef_Name" : predictors, "Coef" : lr_rr.coef_})
lr_rrDf
```

	Coef_Name	Coef
0	2P_O	3.406410
1	3P_O	2.602318
2	FTR	0.216935

```
(ggplot(lr_rrDf, aes(x = "Coef_Name", y = "Coef", fill = "Coef_Name")) + geom_bar(stat = "identity")
+ggtitle("Magnitude of Three Predictor Values from Ridge Regression Model")
+labs(x = "Coefficient Name", y = "Coefficient Value")
+ theme_minimal())
```

Caption: Bar graph of three predictor values for ridge regression model. Two-Point Shot Percentage (2P_O), Three-Point Shot Percentage (3P_0), Free-Throw Rate (FTR).


```
(ggplot(lr_rrDf, aes(x = "Coef_Name", y = "Coef", fill = "Coef_Name")) + geom_point()
+ggtitle("Magnitude of Three Predictor Values from Ridge Regression Model")
+labs(x = "Coefficient Name", y = "Coefficient Value")
+ theme_minimal())
```



Caption: Scatterplot of three coefficient values of ridge regression model. Two-Point Shot Percentage (2P_O), Three-Point Shot Percentage (3P_0), Free-Throw Rate (FTR).

Predicting then Adjusted Offensive Efficiency involved creating a linear regression model. Although not necessary a ridge regression model was used for this question to create a more accurate linear regression model. The coefficient value of Two-Point Shot Percentage was 3.4, the coefficient value of Three-Point Shot Percentage was 2.6. And the Free Throw Rate coefficient value was 0.22.

The two-point shot had the greatest impact on predicting Adjusted Offensive Efficieny and it is logical because the two-point shot is the most common shot in basketball.

QUESTION 2:

- Show model predicted vs true values using all predictors over wins.
- add a ggplot

```
`ggplot: (8774768588729)`
```

```
data2 = pd.read_csv("https://raw.githubusercontent.com/tmoore-byte/392Final/main/cbb21.csv")


predictors2 = ["G", "ADJOE", "ADJDE","BARTHAG", "EFG_O", "EFG_D", "TOR", "TORD",
               "ORB", "DRB", "FTR", "FTRD", "2P_O", "2P_D", "3P_O", "3P_D",
               "ADJ_T", "WAB" , "SEED"]
X_train, X_test, y_train, y_test = train_test_split(data2[predictors2],
                                                    data2["W"], test_size=0.25)

print(X_train.shape)
print(X_test.shape)
```

```
(260, 19)
(87, 19)
```

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
# predictions
y_pred = model.predict(X_test)
y_pred[1:10]
```

```
array([14.13197121,  7.53497344, 12.30369427,  9.05751027, 15.40325506,
       13.65549689,  9.46436849, 20.20436894, 10.41348271])
```

```
model.score(X_test, y_test) #testing R2
```

```
0.8961071750879883
```

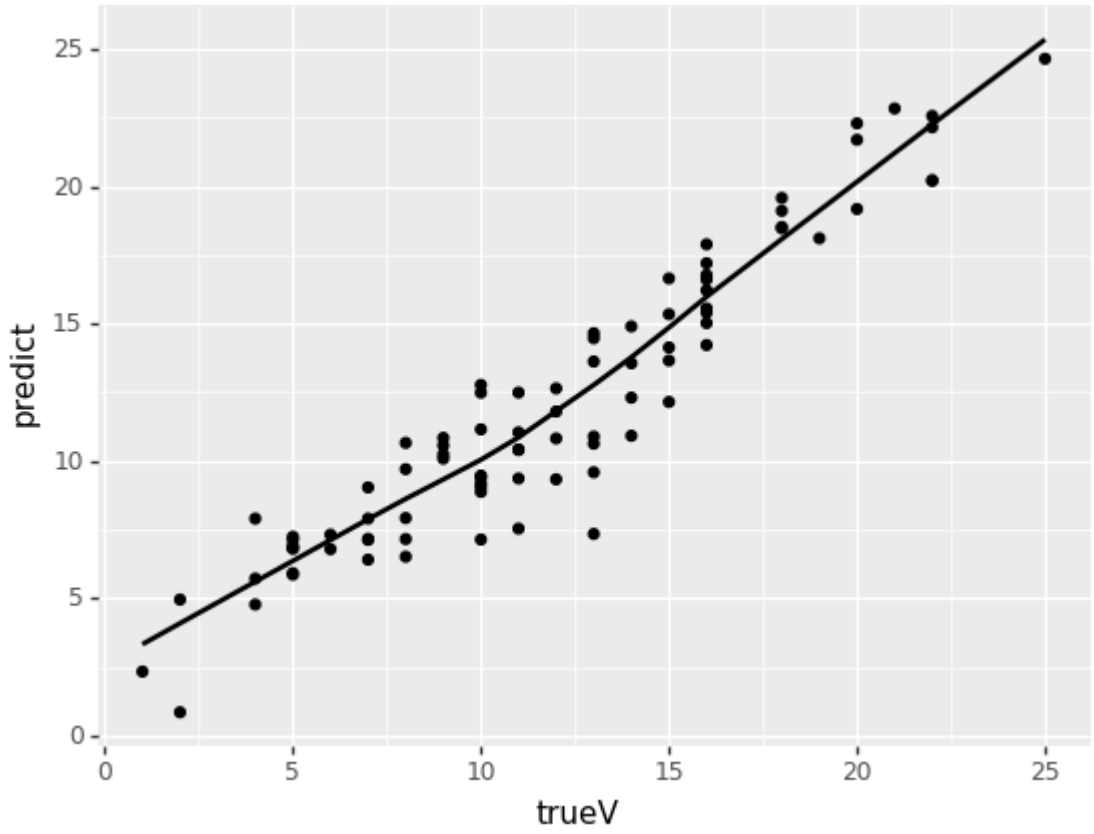
```
model.score(X_train, y_train) #training R2
```

```
0.9053649387756604
```

```
true_vs_pred = pd.DataFrame({"predict": y_pred,"trueV": y_test})
true_vs_pred.head()
```

```
            predict  trueV
```

```
(ggplot(true_vs_pred, aes(x = "trueV", y = "predict")) + geom_point()+geom_smooth())
```




```
<ggplot: (8739169111889)>
```

As seen in the graph above there is not much distance between our true and predicted vlaues in determinign a Winning team with all the predictors. The distance from the line represents the mean error from our predicted values.

```
data2b = pd.read_csv("https://raw.githubusercontent.com/tmoore-byte/392Final/main/cbb21.csv")
```

```
# checking which variables had highest impact on outcome success
features2 = ["G", "ADJOE", "ADJDE","BARTHAG", "EFG_O", "EFG_D", "TOR", "TORD",
```

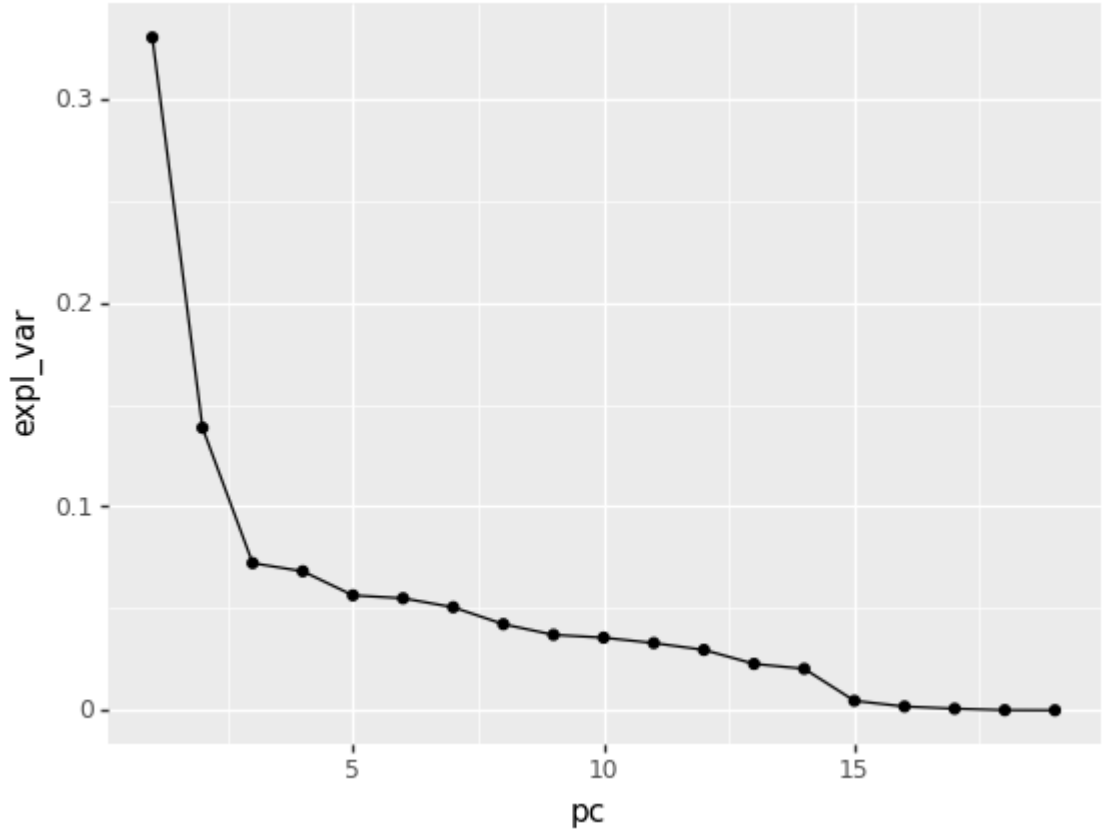
```
               "ORB", "DRB", "FTR", "FTRD", "2P_O", "2P_D", "3P_O", "3P_D",
               "ADJ_T", "WAB" , "SEED"]
z = StandardScaler()
data2b[features2] = z.fit_transform(data2b[features2])
pca = PCA()
pca.fit(data2b[features2])

pcaDF = pd.DataFrame({"expl_var" : pca.explained_variance_ratio_, "pc": range(1,20),
                      "cum_var": pca.explained_variance_ratio_.cumsum(),
                      "name": data2b[features2]})
pcaDF.head()
```

	expl_var	pc	cum_var	name
0	0.329933	1	0.329933	(G,)
1	0.138582	2	0.468515	(A, D, J, O, E)
2	0.072236	3	0.540751	(A, D, J, D, E)
3	0.068228	4	0.608980	(B, A, R, T, H, A, G)
4	0.056448	5	0.665427	(E, F, G, _, O)

as we can see in the data frame above and the corresponding graphs below, games played makes up for 33 % of the variance in our model and then it is adjusted offensive efficiency and adjusted defensive efficiency. These 3 variables are the most important variables when determining number of wins when looking at all predictors.

The first three points in the graph below represent G, ADJOE, ADJDE. All other predictors make up less than .1 of the variance in our model.

```
p1 = (ggplot(pcaDF, aes(x = "pc", y = "expl_var")) + geom_line() + geom_point())


print(p1)
```



<ggplot: (8730159639013)>

QUESTION 3:

- building a model that will most accurately cluster a and b and explaining why not the others

a. Offensive rebound rate (ORB) vs. adjusted tempo (ADJ_T)

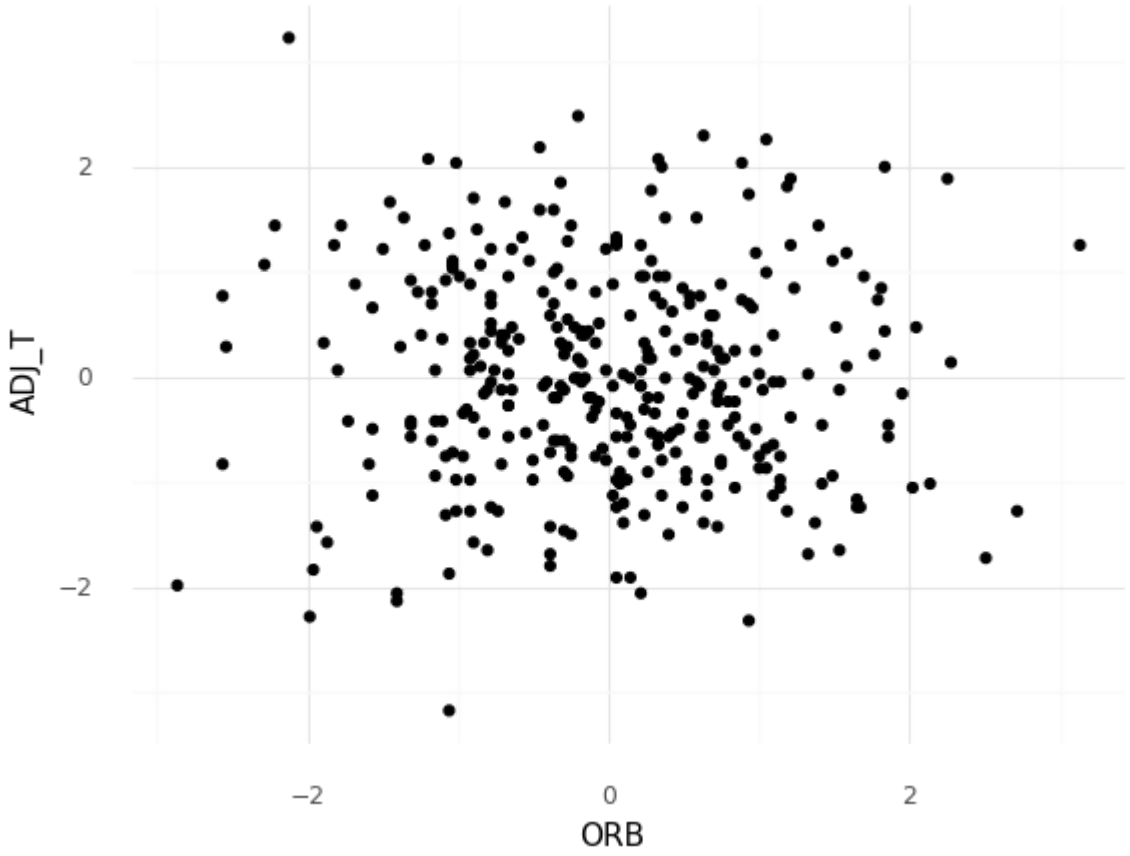
b. Three point percentage (3P_O) vs Effective fg percentage (EFG_O)

```
data3 = pd.read_csv("https://raw.githubusercontent.com/tmoore-byte/392Final/main/cbb21.csv")
data3.head()
```

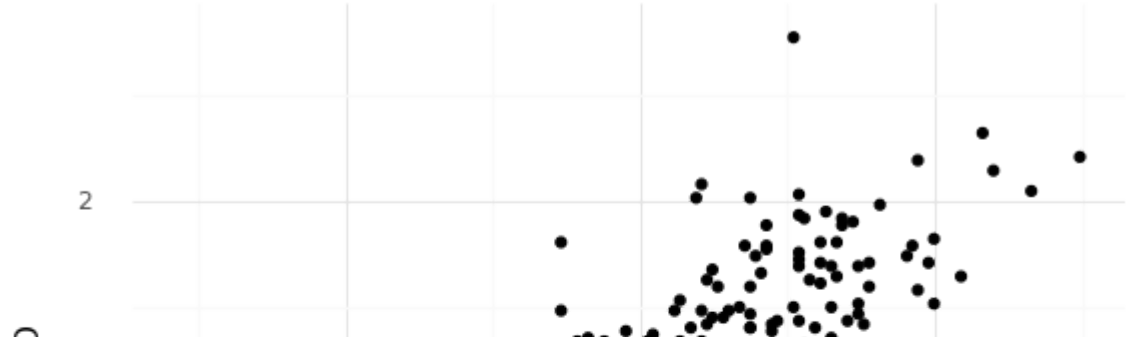
	TEAM	CONF	G	W	ADJOE	ADJDE	BARTHAG	EFG_O	EFG_D	TOR	TORD	ORB	D
0	Michigan	B10	24	20	118.1	91.1	0.9521	54.9	44.9	16.3	15.1	29.4	24
1	Baylor	B12	24	22	123.2	94.5	0.9548	57.5	49.1	17.6	24.6	37.5	30
2	Illinois	B10	29	23	117.7	90.4	0.9539	55.6	46.6	18.2	16.1	33.0	22
3	Gonzaga	WCC	26	26	125.4	89.8	0.9791	61.0	47.5	16.1	20.3	30.4	23
4	Iowa	B10	29	21	123.5	95.7	0.9491	54.6	48.3	13.3	16.3	30.7	28

```
predictors3 = ["ORB", "ADJ_T", "3P_O", "EFG_O"]
X3 = data3[predictors3]
z = StandardScaler()
X3[["ORB", "ADJ_T", "3P_O", "EFG_O"]] = z.fit_transform(X3)


# checking out plotted data
graph_a = (ggplot(X3, aes("ORB", "ADJ_T"))+geom_point()+theme_minimal())
graph_b = (ggplot(X3, aes("3P_O", "EFG_O"))+geom_point()+theme_minimal())
print("GRAPH A: ", graph_a)
print("GRAPH B: ", graph_b)
```

GRAPH A:



<ggplot: (8739175495873)>
GRAPH B:



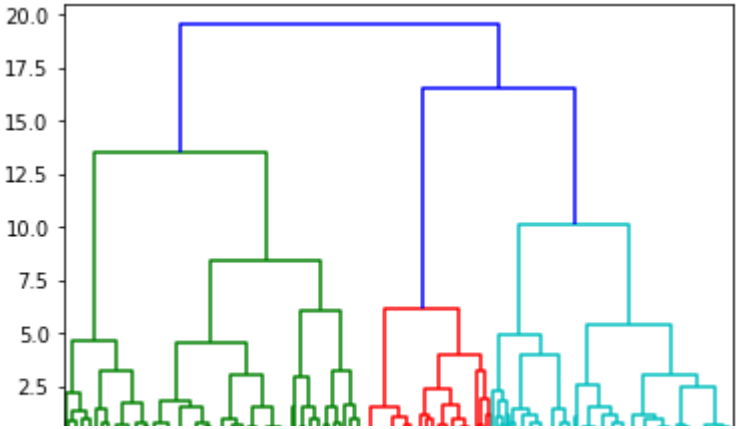
Q3 Graph analysis:

After looking at graph A, I notice that there does not seem to be a linear correlation between adjusted tempo and offensive rebounds. There seems to be a little bit of noise but I am actually interested in seeing what category my model puts these onomilies in, and hopefully I can make an educated guess on the team's schema (how they play). Therefore, I will not be using DBSCAN, and also there seems to be different amounts of density between potential clusters. I am choosing to do **heiarchial cluserting** for **graph A** because it allows us to take clusters and merge them together in a hierarchy. We start with making specific clusters at the bottom of the hierarchy and go all the way up increasing in cluster classification broadness. Although this method is slow in runtime, our data set is small enough to not have any big issues. The flexability of number of clusters and clear adaptable relationships we can set up between clusters makes this very applicable to this data set. One problem is that you cannot unmerge clusters.

When looking at graph B, I notice a very strong positive linear correlation between the variables associated with effective fg% and 3 point % on offense. The overall point variance seems relatively uniform but maybe a little lower variance right around the mean on each axis. There is some noise but it is not too noticeable with the naked eye.

For graph BI will be implementing a **DBSCAN model** here because I am interested in seeing how we can deal with noise and since we have a very simple graph with only two predictors, our model should pick up our clusters effectively. I just need to look out for point overlapping since most of the data is very close together. I do not expect this model to perform that well with the given predictors but it will be interesting to see how it ends up grouping the points.

```
#heiarchial cluserting graph a
features3 = ["ORB", "ADJ_T"]
x3 = data3[features3]
z = StandardScaler()
x3[features3] = z.fit_transform(x3) # z scoring


# ward allows for different variance clustering, distcance metric = ward
hac = AgglomerativeClustering(affinity = "euclidean", linkage = "ward")
hac.fit(x3)
dendrogram = sch.dendrogram(sch.linkage(x3, method = 'ward'))
```




```
hac = AgglomerativeClustering(n_clusters = 3, # gave me the best score
 affinity = "euclidean",
 linkage= "ward")
hac.fit(x3)
membership = hac.labels_
membership
```

```
array([1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 2, 1, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 2, 1, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 2, 0, 1, 0, 2, 0, 1, 1, 1, 1, 1,
       0, 1, 2, 0, 1, 1, 1, 2, 2, 2, 2, 0, 0, 0, 1, 2, 1, 0, 0, 1, 2, 2,
       1, 1, 2, 1, 1, 0, 2, 0, 2, 1, 2, 2, 2, 2, 2, 0, 2, 1, 2, 2, 0, 1,
       2, 0, 2, 2, 0, 0, 0, 2, 1, 1, 0, 0, 1, 0, 0, 1, 2, 2, 0, 0, 1, 2,
       0, 0, 0, 0, 2, 1, 2, 2, 2, 0, 1, 0, 1, 0, 2, 1, 1, 0, 1, 1, 1, 0,
       2, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 2, 2, 2, 2, 1, 2, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 2, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 2, 0, 0, 1, 1, 2, 0, 1, 0, 1, 2,
       2, 2, 1, 1, 0, 1, 2, 0, 0, 1, 0, 0, 0, 2, 1, 2, 1, 0, 1, 0, 2, 1,
       0, 2, 1, 1, 1, 1, 1, 1, 1, 0, 2, 0, 2, 0, 0, 0, 0, 0, 1, 1, 2, 2,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 2, 1, 1, 0, 0, 0, 1, 0, 2, 2, 0, 1,
       0, 2, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       2, 1, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 1, 1, 2, 2, 0, 0,
       1, 0, 2, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])
```

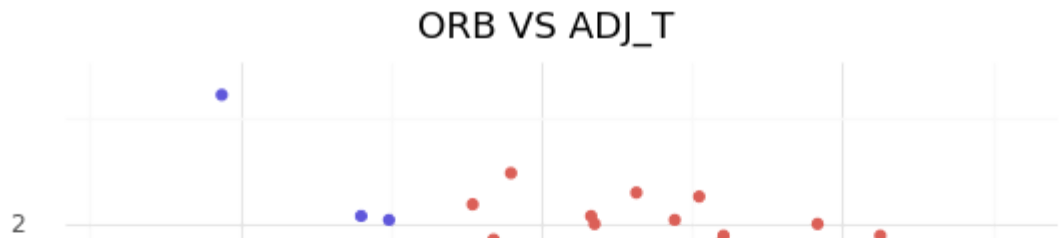
```
print("silhouette score: ")
```

```
silhouette_score(x3,membership)
```

```
     silhouette score:
     0.26451992069649716
```

As we can see in graph a, our heirarchial clustering model does not produce clusters effectively. We would want to see our score up in the .75 or higher range to consider our model accurate at clustering teams based on offensive rebounding and adjusted tempo.

```
# re making the graph A
# re-making Sugar vs Total Fat
x3["cluster"] = membership
(ggplot(x3, aes(x = "ORB", y = "ADJ_T", color = "factor(cluster)"))+ geom_point() +
theme_minimal() + ggtitle("ORB VS ADJ_T"))
```



Along the x axis, anything above 0 indicates a team with higer than average offensive rebounding rates, and anything lower represents lower than average offensive rebounding rates within the dataset. The same goes for the y axis, higher/lower than 0 values of adjusted tempo represents higher/lower adjusted tempo rates based on the average values per team in this data set.

With this graph we can see our h clustering model broke up teams into 3 categories. We have teams with **low offensive rebounding rates** and **mid level adjusted tempo (green)**. This cluster could represent teams that have average to smaller size players (since they do not get many rebounds on the offensive end) and they could either play at a slower pace or they could allow more offensive rebounds to the opposition (hence the low amount of possessions indicated by the less than 0 value on the ADJ_T axis)

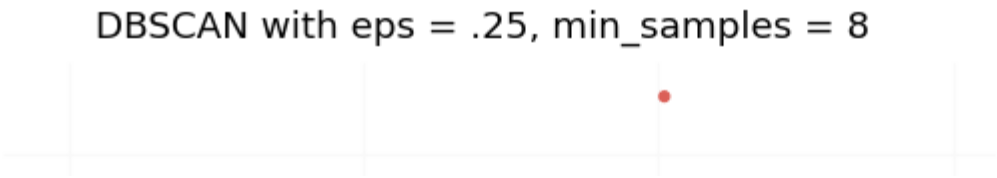
The **blue** cluster has **higher adjusted tempo** than the average team in this dataset and **slightly less than average ORB**. This cluster could represent teams that do not hustle on the offensive glass but focus more on getting more possesions per game than the average team. This could mean getting more steals, shooting more often, etc.

The **red** cluster has **higher to mid adjusted tempo** and **higher offensive rebounds** than the average team in this dataset. This could indicate that a team in the red cluster has dominant bigs (centers and powerforwards) that get more than the average teams' amount of rebounds. The red cluster teams have a bit more dispersed ajusted tempo and this makes sense because of the different way teams use their big men. Big men can either be used to initiate offense (indicating lower adjusted tempo since offense comes slower and teams use more of the shot clock) or the teams could use their bigs as rebounders and they can pass the ball out on the fast-break to initiate fast paced offense (this would indicate the red teams with higher adjusted tempo).

Overall, this graph is very telling of how a team plays even though the score is not very high. This visaul could be used as a generalization for entry level basketball analysis to get a better understanding of how a team likes to use their players in their schema (although it is only speculation. The only thing we are actually predicting are rebounding and tempo).

```
features3b = ["3P_O", "EFG_O"]
x3b = data3[features3b]
```

```
z = StandardScaler()
x3b[features3b] = z.fit_transform(x3b) # z scoring

# creating the DBSCAN algorithm
db = DBSCAN(eps = 0.25, min_samples = 8).fit(x3b)
labsList = ["Noise"]
labsList = labsList +["Cluster " + str(i) for i in range(1,len(set(db.labels_)))]
#ceating assignments
x3b["assignments"] = db.labels_
#plotting
(ggplot(x3b, aes(x = "3P_O", y = "EFG_O", color = "factor(assignments)")) +
geom_point() +
theme_minimal() +
scale_color_discrete(name = "Cluster Assignment",
 labels = labsList) +
theme(panel_grid_major = element_blank()) +
labs(title = "DBSCAN with eps = .25, min_samples = 8"))
```




```
# calculating silhouette score, only looking at non-noise
X_clustered = x3b.loc[x3b.assignments >=0]
print("DBSCAN silhouette score: ")
print(silhouette_score(X_clustered[["3P_O","EFG_O"]], X_clustered["assignments"]))
```

```
DBSCAN silhouette score:
0.27928211753912663
```

Our silhouette score in our DBSCAN Algorithm is around the same in terms of performance in clustering efficiency as our model in part a of this question. We can see there is a larger amount of noise with our given parameters than I initially predicted, but overall there seems to be some strong correlations between clusters at a glance.

With the 5 main clusters (yellow, green, lightblue, purple, and pink), the teams with higher than 0 3P_O (part of yellow cluster and)

QUESTION 4:

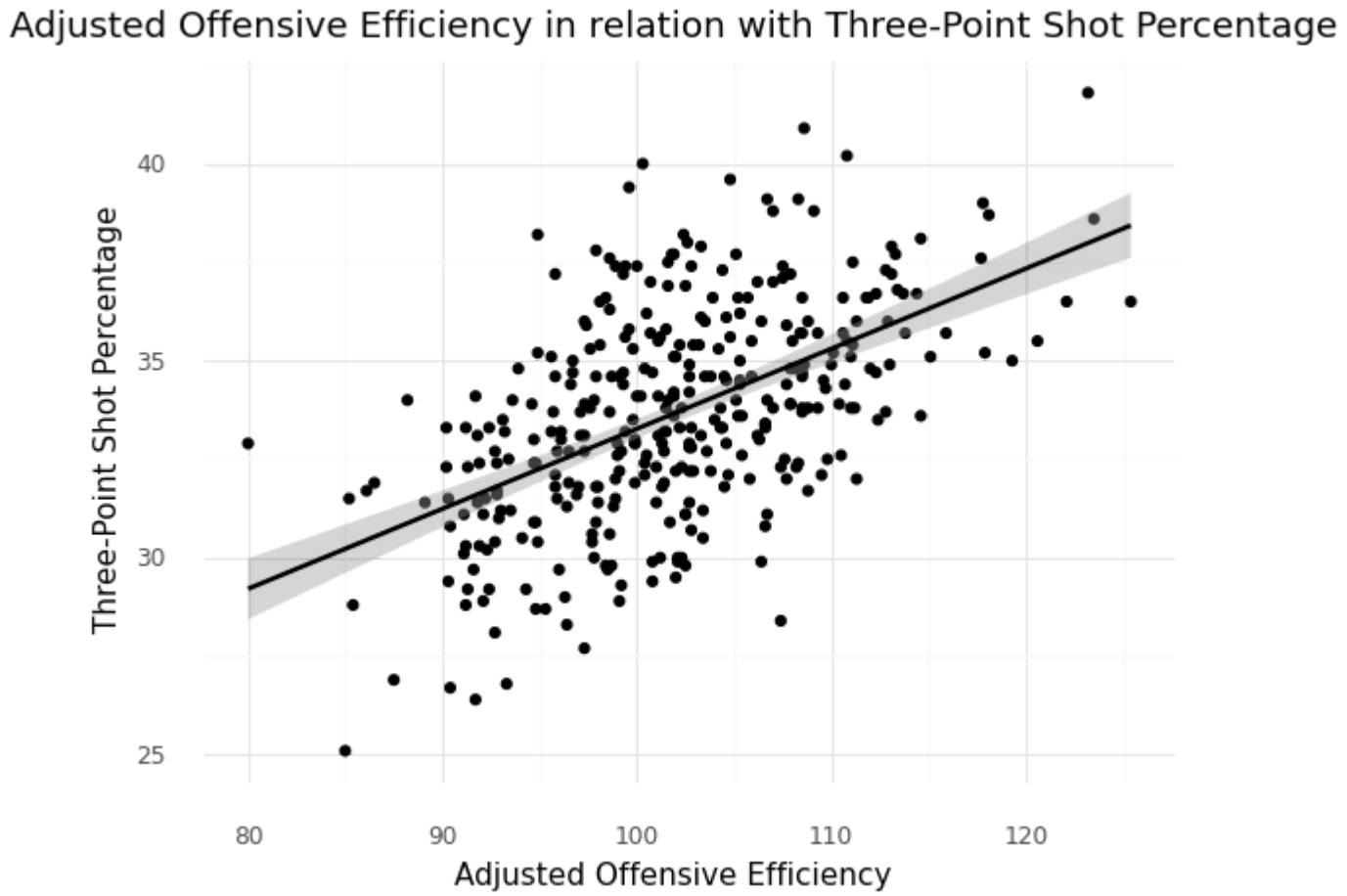
- What is the relationship between Adjusted Offensive Efficiency (ADJOE) and Three-Point Shot Percentage (3P_O)? What does the relationship explain, and how many clusters need to be created?

```
km = KMeans(n_clusters = 4)
km.fit(X)

membership = km.predict(X)
X["clusters"] = membership
```

```
silhouette_score(X, membership)
```

```
0.4420675726720011
```

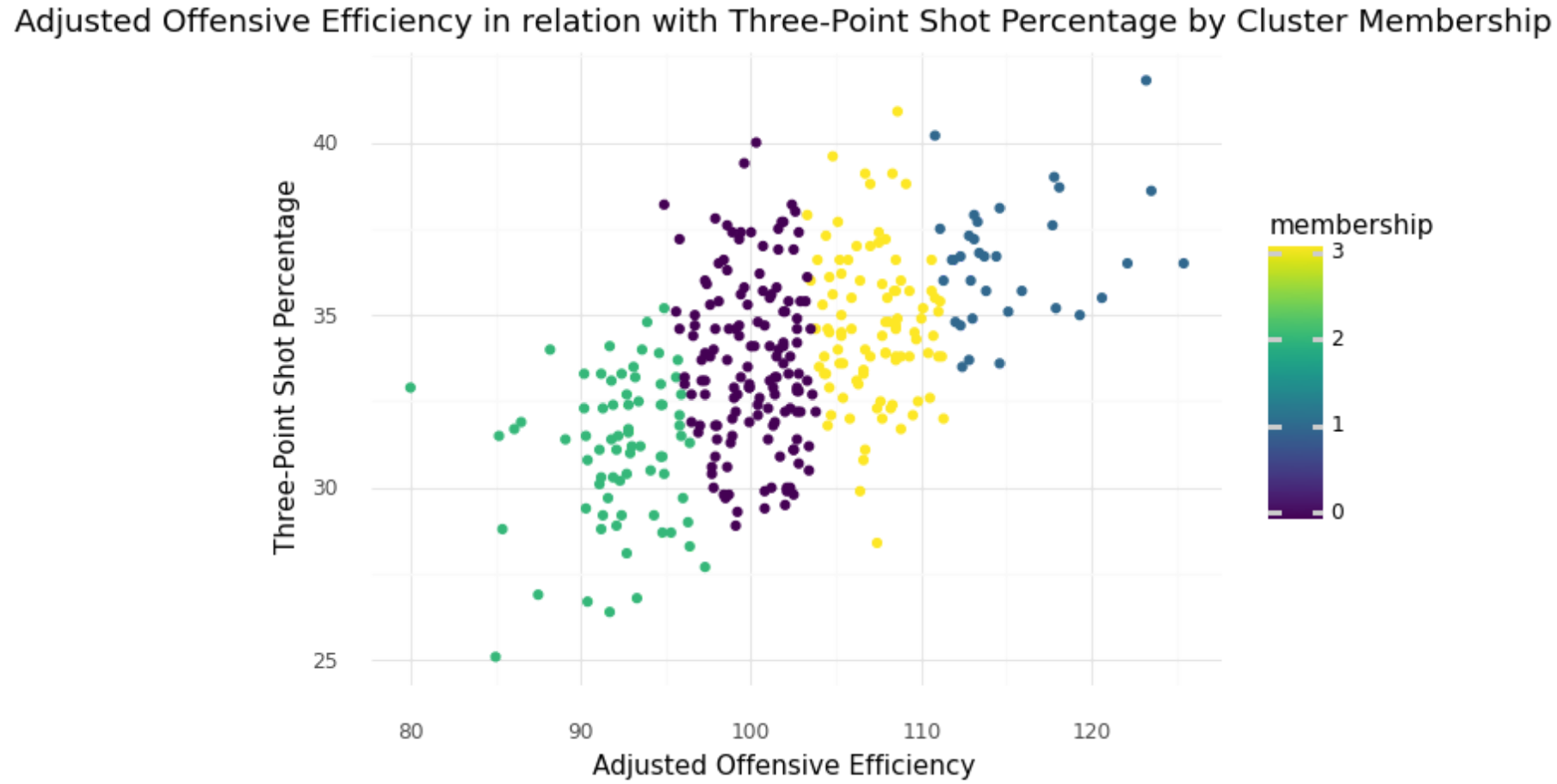
```
(ggplot(X, aes(x = "ADJOE", y = "3P_O")) + geom_point() + stat_smooth(method = "lm")
+ ggtitle("Adjusted Offensive Efficiency in relation with Three-Point Shot Percentage")
+ labs(x = "Adjusted Offensive Efficiency", y = "Three-Point Shot Percentage")
+ theme_minimal())
```




```
<ggplot: (8782511947289)>
```

Caption: The ggplot above shows the relationship between Adjusted Offensive Efficiency and Three-Point Shot Percentage. There is a linear relationship between both variables as Adjusted Offensive Efficiency increases, the Three-Point Shot Percentage increases.

```
(ggplot(X, aes(x = "ADJOE", y = "3P_O", color = "membership")) + geom_point()
+ ggtitle("Adjusted Offensive Efficiency in relation with Three-Point Shot Percentage by Cluster Membership")
+ labs(x = "Adjusted Offensive Efficiency", y = "Three-Point Shot Percentage")
+ theme_minimal())
```




```
<ggplot: (8782495839121)>
```

Caption: The ggplot above shows the relationship between Adjusted Offensive Efficiency and Three-Point Shot Percentage. The data

Answer to question 4: There is a linear relationship between Adjusted Offensive Efficiency and Three-Point Shot Percentage. As Adjusted Offensive Efficiency increases the Three-Point Shot Percentage increases, and this is logical because a highly efficient offense will also shoot three-point shots at a high percentage. For this relationship I chose to divide the data into four clusters. The reasoning behind this is that the March Madness Tournament assigns teams a ranking between 1 to 16. So each cluster represents teams who can be ranked between 1 to 4, 5 to 8, 9 to 12, and 13 to 16.

As shown in the first ggplot for question 4, the data is linear and has low seperation besides a few outliers. The shape of the data is like a diagonal rectangle and in order to assign the data to circular clusters, the K-Means algorithm was used. The model produced a silhouette score of 0.44 which means there is strong cohesion and sufficient separation. However, the few outliers in the green and blue clusters lowers the silhouette score.

To give basketball context of the clusters, the green cluster representing teams ranked between 13 to 16 are Ohio, Drexel, Eastern Washington etc. The purple cluster representing teams ranked between 9 to 12 are Wisconsin, UCLA, Georgetown, etc. The yellow cluster representing teams between 5 to 8 are Villanova, Texas Tech, Florida, etc. Lastly, the blue cluster representing teams ranked 1 to 4 are Gonzaga, West Virgina, Kansas, etc.

QUESTION 5:

- Using only Number of Games Played (G) and offensive factors such as, Adjusted Offensive Efficiency (ADJOE), Effective Field Goal Percentage Shot (EFG_O), Turnover Percentage Committed (TORD), Offensive Rebound Rate (ORB), Free Throw Rate (FTR), Two-point Percentage Shot (2P_O), Three-point Percentage Shot (3P_O), Adjusted Tempo (ADJ_T) predict the number of wins for a team. How accurate is the model?

```
predictors5 = ['G', 'ADJOE', 'EFG_O','TORD', 'ORB',  'FTR',  '2P_O',  '3P_O', 'ADJ_T']
X = data[predictors5]
y = data["W"]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

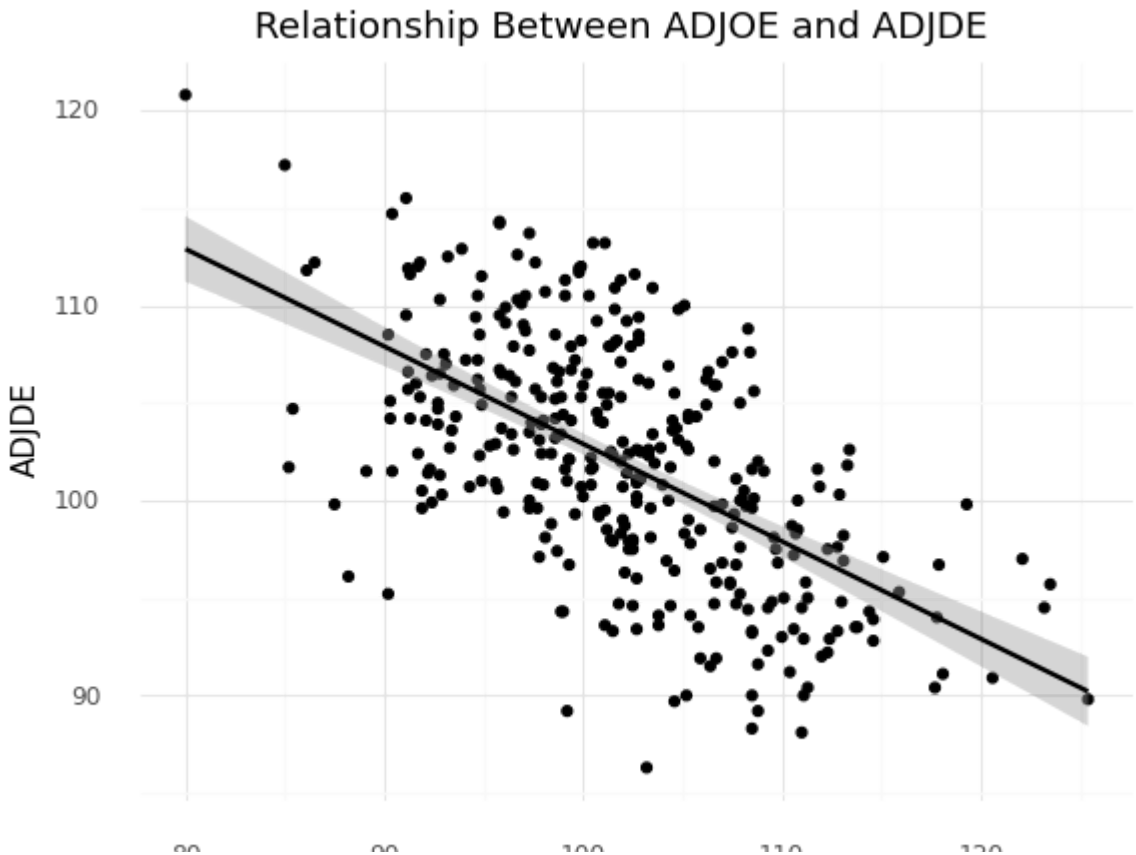
```
z = StandardScaler()
z.fit(X_train)
Xz_train = z.transform(X_train)
Xz_test = z.transform(X_test)

model = LinearRegression()
model.fit(Xz_train, y_train)
```

```
     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
test_pred = model.predict(Xz_test)
train_pred = model.predict(Xz_train)


test_mse = mean_squared_error(y_test, test_pred)
train_mse = mean_squared_error(y_train, train_pred)

test_r2 = r2_score(y_test, test_pred)
train_r2 = r2_score(y_train, train_pred)


print("Testing MSE: " , test_mse)
print("Train MSE: " , train_mse)
print("Test r2: " , test_r2)
print("Train r2: " , train_r2)
```

```
     Testing MSE:  9.029909934394519
     Train MSE:  8.664108048819765
     Test r2:  0.6534646414141323
     Train r2:  0.6923642776291535
```

```
(ggplot(data, aes (x = "ADJOE", y = "ADJDE")) + geom_point() + stat_smooth(method = "lm")
+ ggtitle("Relationship Between ADJOE and ADJDE")
+ theme_minimal())
```

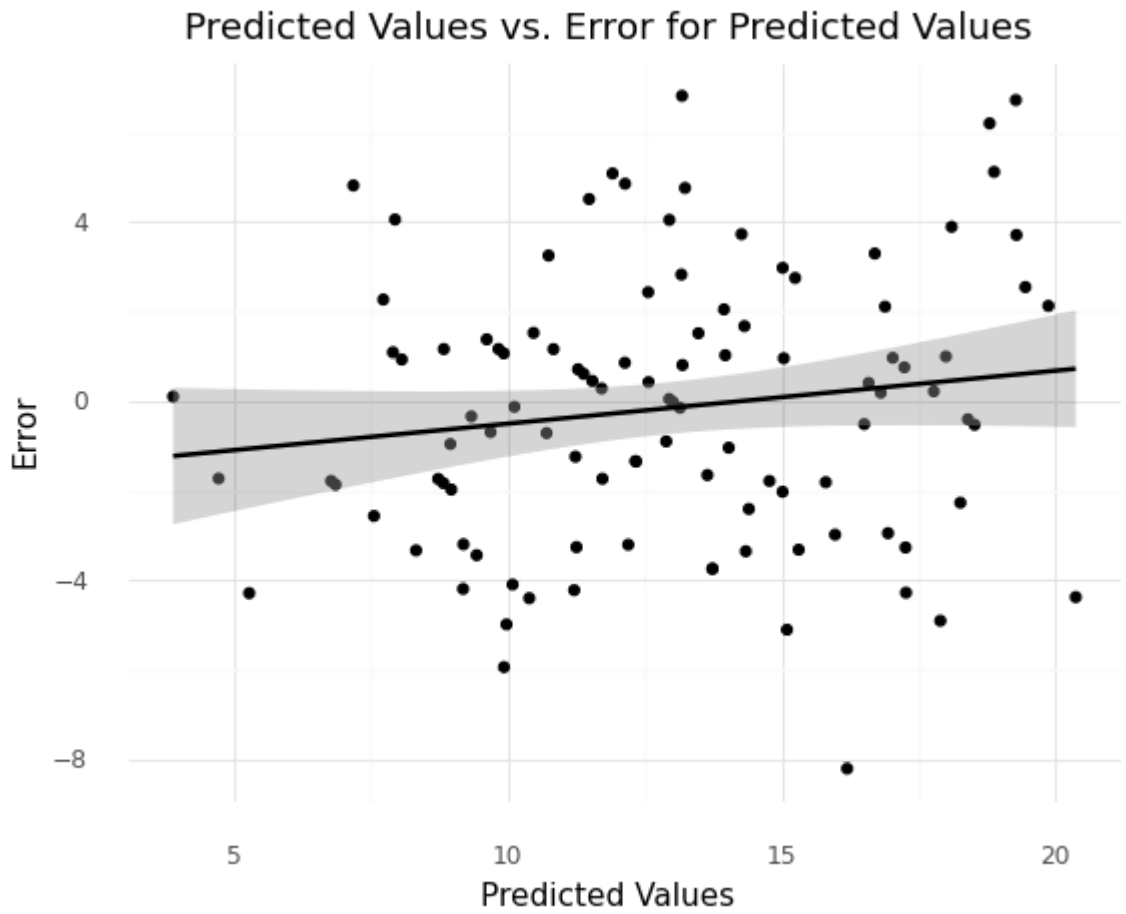


Caption: The ggplot above shows the relationship between Adjusted Offensive Efficiency and Adjusted Defensive Efficiency. There is an inverse linear relationship between ADJOE and ADJDE.

```
assump = pd.DataFrame({"Error": y_test - test_pred, "Predicted": test_pred, "True" : y_test})
assump.head()
```

	Error	Predicted	True
96	-1.856291	6.856291	5

```
(ggplot(assump, aes(x = "Predicted", y = "Error")) + geom_point() + stat_smooth(method = "lm")
+ ggtitle("Predicted Values vs. Error for Predicted Values")
+ labs(x = "Predicted Values", y = "Error")
+ theme_minimal())
```




```
<ggplot: (8782493910077)>
```

Caption: The ggplot above shows the predicted values compared to the error from the model and it represents the spread of the error. The ggplot also follows the assumption of homoscedasticity because the spread of the error is equal throughout the scatterplot.

There is also a linear relationship between the predicted values and the error.

Answer to question 5: Using only number of games played and offensive factors to predict the number of wins for a team shows that some defensive factors are needed to build a more accurate model. The model accounts for 69% of variance on the training set and 65% on the testing set. Although this is not the highest r2 score, the model is still performing well at predicting number of wins using only offensive factors.

By looking at the first ggplot for this question, the relationship between ADJOE and ADJDE show that teams who are better on offense are not as good at defense and teams that a better at defense are not as good at offense. And then there are teams which have somewhat of a balance between offense and defense.

In the second ggplot for this question the spread of the error is equally distributed across the scatterplot which is fine for this model because it follows the assumption of homoscedasticity. However, the spread of the error also shows how sometimes the model is very close at predicting the number of wins and sometimes it is really off by either predicting too low or too high.

For example, the model predicted 18 wins for a team that had 18 wins, but at the same time the model predicted 19 wins for a team but the team actually had 26 wins. The team that was predicted to have 19 wins likely had lower offensive statistics and high defensive statistics because their actual number of wins was 26.

Although the model is still performing fairly well with even error spread throughout the data, there are limitations and it shows that defensive statistics are needed to accurately predict a team's number of wins.

QUESTION 6:

- Based on all variables (expect Power Rating (BARTHAG)), which variables are most important in predicting Power Rating (BARTHAG)?

```
features = ['W', 'ADJOE', 'ADJDE', 'EFG_O', 'EFG_D',
       'TOR', 'TORD', 'ORB', 'DRB', 'FTR', 'FTRD', '2P_O', '2P_D', '3P_O',
       '3P_D', 'ADJ_T', 'WAB', 'SEED']


X = data[features]
y = data['BARTHAG']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)

z = StandardScaler()
X_train[features] = z.fit_transform(X_train[features])
X_test[features] = z.fit_transform(X_test[features])


model = LinearRegression()
model.fit(X_train, y_train)

print("Train: " , mean_absolute_error(y_train, model.predict(X_train)))
print("Test: " , mean_absolute_error(y_test, model.predict(X_test)))
```

```
     Train:  0.03305216259835463
     Test:  0.03926608668849894
```

```
rr = Ridge()

rr.fit(X_train, y_train)

print("Train: " , mean_absolute_error(y_train, rr.predict(X_train)))
print("Test: " , mean_absolute_error(y_test, rr.predict(X_test)))
```

```
     Train:  0.03324125222609486
     Test:  0.03864982749098832
```

```
lsr = Lasso()

lsr.fit(X_train,y_train)

print("Train: " , mean_absolute_error(y_train, lsr.predict(X_train)))
print("Test: " , mean_absolute_error(y_test, lsr.predict(X_test)))
```

```
     Train:  0.22332137490608564
     Test:  0.22593935064935064
```

```
rrDf = pd.DataFrame({"Coef_Name" : features, "Coef" : rr.coef_})
```

```
rrDf.head()
```

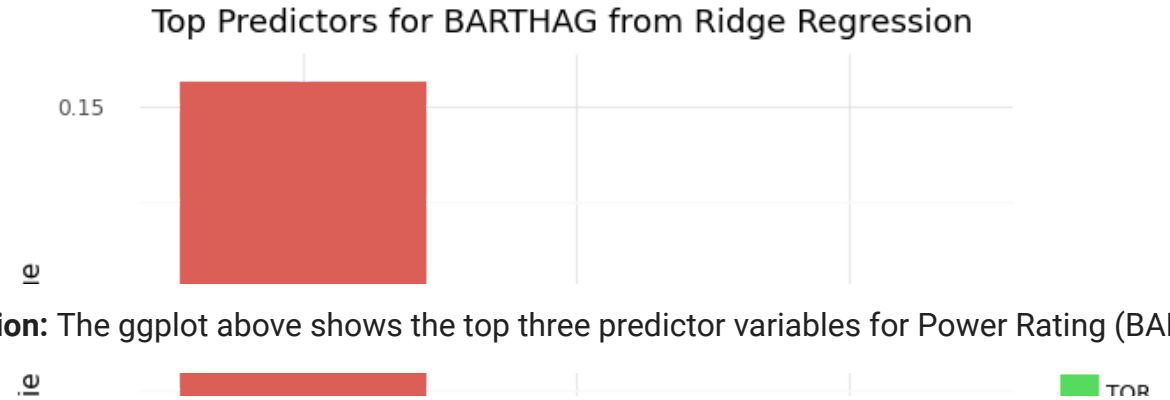
	Coef_Name	Coef
0	W	0.007061
1	ADJOE	0.156577
2	ADJDE	-0.112242
3	EFG_O	-0.003789
4	EFG_D	-0.010975

```
(ggplot(rrDf, aes(x = "Coef_Name", y = "Coef", fill = "Coef_Name")) + geom_point()
+ ggtitle("Scatter Plot of Coefficient Values from Ridge Regression")
+ labs(x = "Coefficient Name", y = "Coefficient Value")
+ theme_minimal())
```

Caption: The ggplot above shows the spread of the coefficient values in the Ridge Regression model.

```
input = [["ADJOE", 0.156577], ["WAB", 0.029759], ["TOR", 0.015771]]
rrDf_input = pd.DataFrame(input, columns = ["Coef_Name", "Coef"])
```

```
(ggplot(rrDf_input, aes(x = "Coef_Name", y = "Coef", fill = "Coef_Name")) + geom_bar(stat = "identity")
+ ggtitle("Top Predictors for BARTHAG from Ridge Regression")
+ labs(x = "Coefficient Name", y = "Coefficient Value")
+ theme_minimal())
```



Caption: The ggplot above shows the top three predictor variables for Power Rating (BARTHAG) from the Ridge Regression model.

Answer to question 6: To evaluate which variables are most important in predicting Power Rating (BARTHAG) a linear regression model was built.

The original linear regression model produced a mean absolute error of 0.033 on the training set and 0.039 on the testing set. The ridge regression model produced a mean absolute error of 0.033 on the training set and 0.039 on the testing set. Lastly, the lasso regression model produced a training score of 0.223 on the training set and 0.226 on the testing set.

Based on the three models the lasso regression model showed lower signs of overfitting to the training set compared to the other two models, but it also had a higher mean absolute error score.

Since ridge regression does not reduce coefficient values to zero like lasso regression does, the coefficient values of the ridge regression model were used to identify the three predictor variables for Power Rating (BARTHAG).

The top three predictors were Adjusted Offensive Efficiency (ADJOE), Turnover Percentage Allowed (TOR), and Wins Above Bubble (WAB). These are the most logical top predictors for Power Rating (BARTHAG) because the teams most likely to beat an average division 1 team are likely to have a highly efficient offense, low turnover rate, and a high number of wins to qualify for the March Madness tournament.