NBA_Kearns_Analysis

May 18, 2022

[ ]:
```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from plotnine import *

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier # Decision Tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.linear_model import LogisticRegression # Logistic Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables

from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
from sklearn.cluster import KMeans
from sklearn.model_selection import LeaveOneOut #LOO cv
from sklearn.model_selection import cross_val_score # cross validation metrics
from sklearn.model_selection import cross_val_predict # cross validation metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score␣
↪#model evaluation
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from sklearn.model_selection import GridSearchCV
%precision %.7g
%matplotlib inline
```

1 Cleaning The Data

[ ]:
```
nba = pd.read_csv("https://query.data.world/s/fwbezkloolkpuahajrtzju5dxthw4p")
nba.columns
nba.drop("Unnamed: 0",axis = 1)
```

[ ]:
```
      Team  Game        Date  Home Opponent WINorLOSS  TeamPoints  \
0      ATL     1  2014-10-29  Away      TOR         L         102
1      ATL     2  2014-11-01  Home      IND         W         102
2      ATL     3  2014-11-05  Away      SAS         L          92
3      ATL     4  2014-11-07  Away      CHO         L         119
4      ATL     5  2014-11-08  Home      NYK         W         103
…      …     …         …     …      …         …          …
9835   WAS    78  2018-04-03  Away      HOU         L         104
9836   WAS    79  2018-04-05  Away      CLE         L         115
9837   WAS    80  2018-04-06  Home      ATL         L          97
9838   WAS    81  2018-04-10  Home      BOS         W         113
9839   WAS    82  2018-04-11  Away      ORL         L          92

      OpponentPoints  FieldGoals  FieldGoalsAttempted  …  Opp.FreeThrows  \
0                109          40                   80  …              27
1                 92          35                   69  …              18
2                 94          38                   92  …              27
3                122          43                   93  …              20
4                 96          33                   81  …               8
…                …           …                  …  …             …
9835             120          38                   72  …              18
9836             119          47                   94  …              22
9837             103          35                   87  …              16
9838             101          41                   83  …              22
9839             101          33                   95  …              22

      Opp.FreeThrowsAttempted  Opp.FreeThrows.  Opp.OffRebounds  \
0                          33            0.818               16
1                          21            0.857               11
2                          38            0.711               11
3                          27            0.741               11
4                          11            0.727               13
…                         …              …               …
9835                       27            0.667               10
9836                       28            0.786                5
9837                       23            0.696                7
9838                       27            0.815               13
9839                       27            0.815                6

      Opp.TotalRebounds  Opp.Assists  Opp.Steals  Opp.Blocks  Opp.Turnovers  \
0                    48           26          13           9              9
```

```
1                    44          25         5          5            18
2                    50          25         7          9            19
3                    51          31         6          7            19
4                    44          26         2          6            15
…                  …          …         …         …            …
9835                 46          26        13          3             9
9836                 35          26        10          3            16
9837                 50          24         5          5            18
9838                 44          22        14          1            16
9839                 42          20         6          7            16

      Opp.TotalFouls
0                 22
1                 26
2                 15
3                 30
4                 29
…               …
9835              14
9836              14
9837              22
9838              18
9839              27

[9840 rows x 40 columns]
```

2 Q1

```
[ ]: predictors = ['TeamPoints', 'OpponentPoints', 'FieldGoals',␣
      ↪'FieldGoalsAttempted',
             'FieldGoals.', 'X3PointShots', 'X3PointShotsAttempted', 'X3PointShots.',
             'FreeThrows', 'FreeThrowsAttempted', 'FreeThrows.', 'OffRebounds',
             'TotalRebounds', 'Assists', 'Steals', 'Blocks',
             'TotalFouls', 'Opp.FieldGoals', 'Opp.FieldGoalsAttempted',
             'Opp.FieldGoals.', 'Opp.3PointShots', 'Opp.3PointShotsAttempted',
             'Opp.3PointShots.', 'Opp.FreeThrows', 'Opp.FreeThrowsAttempted',
             'Opp.FreeThrows.', 'Opp.OffRebounds', 'Opp.TotalRebounds',
             'Opp.Assists', 'Opp.Steals', 'Opp.Blocks', 'Opp.Turnovers',
             'Opp.TotalFouls']
```

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(nba[predictors],␣
      ↪nba["Turnovers"], test_size=0.2)
     X_train.head()

     z = StandardScaler()
```

```
X_train[predictors] = z.fit_transform(X_train[predictors])
X_test[predictors] = z.transform(X_test[predictors])

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

y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)

#Test Set
r2_test = r2_score(y_test, y_test_pred)
mse_test = mean_squared_error(y_test, y_test_pred)

#Train Set
r2_train = r2_score(y_train, y_train_pred)
mse_train = mean_squared_error(y_train, y_train_pred)
```

[ ]:
```
y_pred = lr.predict(X_test)
true_vs_pred = pd.DataFrame({"predicted": y_pred,
                             "true": y_test})

true_vs_pred
```

[ ]:

	predicted	true
2418	12.038371	13
377	13.627833	12
5057	20.551980	22
7611	11.943297	11
7657	10.887244	9
…	…	…
4624	19.352234	16
9012	11.380040	10
5841	9.855715	9
4747	11.232043	11
1372	21.092919	23

[1968 rows x 2 columns]

[ ]:
```
print("R2 Train: ", r2_train)
print("MSE Train: ",mse_train)

print("R2 Test: ", r2_test)
print("MSE Test: ",mse_test)
```

```
R2 Train:  0.8079776540887771
MSE Train:  2.8690144154489214
R2 Test:  0.8103752014594491
```

MSE Test: 2.8623913142213544

[ ]:
```
z = StandardScaler()
nba[predictors] = z.fit_transform(nba[predictors])

pca = PCA()
pca.fit(nba[predictors])
pcaDF = pd.DataFrame({"expl_var" :
                      pca.explained_variance_ratio_,
                      "pc": range(1,34),
                      "cum_var":
                      pca.explained_variance_ratio_.cumsum()})

pcaDF
(ggplot(pcaDF, aes(x = "pc", y = "cum_var")) + geom_line(color = "pink") +
 geom_point(color = "pink") + theme_minimal() +geom_hline(yintercept = 0.90)+
 ↪labs(title = "Explained Cum Variance Plot"))
```

[ ]: <ggplot: (8734511131585)>

Figure 1.

Explained cumulative variance plot showing the accumulated explainable variance per additional prinicipal component.

```
[ ]: pcomps10 = pca.transform(nba[predictors])
     pcomps10 = pd.DataFrame(pcomps10[:, 0:14])

     #modeMod1
     lr1 = LogisticRegression()
     lr1.fit(nba[predictors], nba["Turnovers"])
     print("all data: ", lr1.score(nba[predictors], nba["Turnovers"]))

     #modeMod1
     lr2 = LogisticRegression()
     lr2.fit(pcomps10, nba["Turnovers"])
     print("14 PCs:   ", lr2.score(pcomps10, nba["Turnovers"]))
```

```
all data:  0.26148373983739837
14 PCs:    0.19654471544715446
```

```
[ ]: pca_y_pred = lr1.predict(X_test)
     pca_true_vs_pred = pd.DataFrame({"predicted": pca_y_pred,
                                      "true": y_test})
```

```
[ ]: pca_train = pca.transform(X_train)
     pca_train = pd.DataFrame(pca_train[:,0:14])

     pca_test = pca.transform(X_test)
     pca_test = pd.DataFrame(pca_test[:,0:14])

     lr.fit(pca_train, y_train)

     pca_train_pred = lr.predict(pca_train)
     pca_test_pred = lr.predict(pca_test)

     #Train
     pca_train_r2 = r2_score(y_train, pca_train_pred)
     pca_train_mse = mean_squared_error(y_train, pca_train_pred)

     #Test
     pca_test_r2 = r2_score(y_test, pca_test_pred)
     pca_test_mse = mean_squared_error(y_test, pca_test_pred)
```

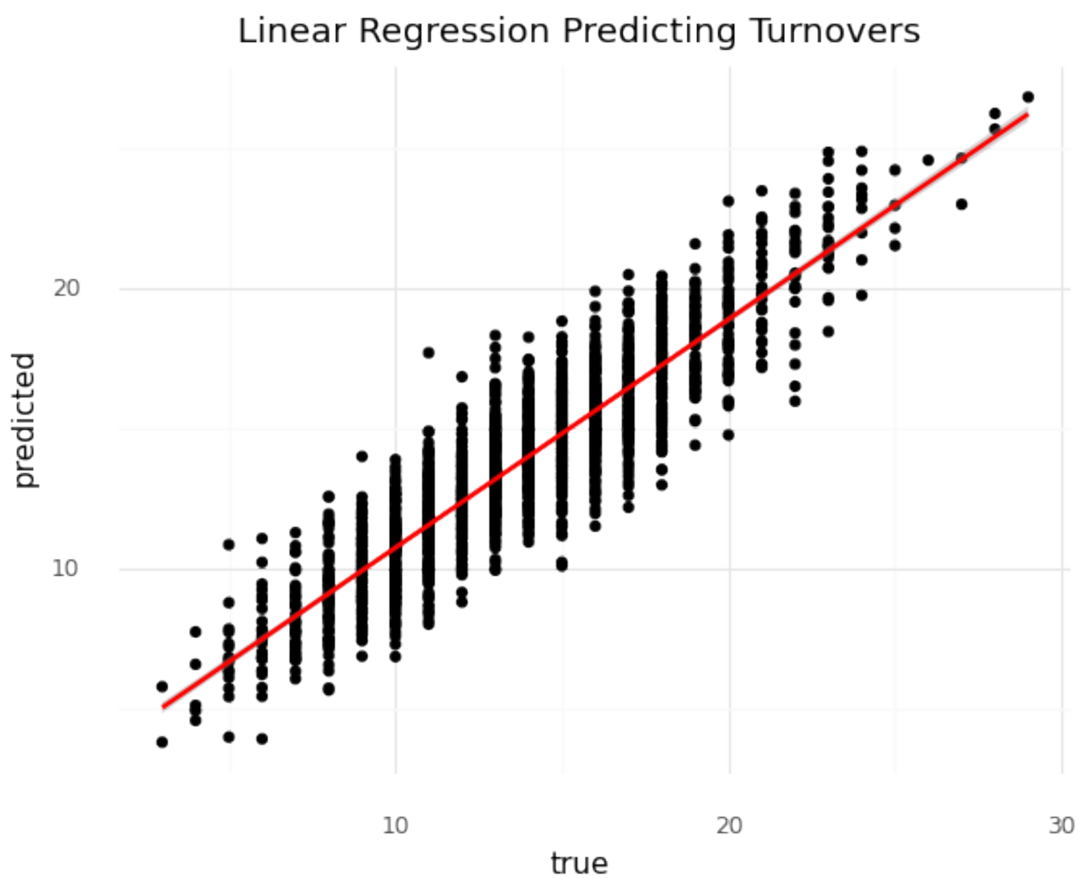
```
[ ]: print("R2 Train: ", pca_train_r2)
     print("MSE Train: ",pca_train_mse)

     print("R2 Test: ", pca_test_r2)
```

```
print("MSE Test: ",pca_test_mse)
```

```
R2 Train:  0.6755809648666886
MSE Train:  4.8471592409032285
R2 Test:  0.6850222470693779
MSE Test:  4.754597452973796
```

Linear Regression Model

```
[ ]: (ggplot(true_vs_pred, aes(x = "true", y = "predicted")) + geom_point() +␣
     ↪theme_minimal() + geom_smooth(method = "lm", color = "red") + labs(title =␣
     ↪"Linear Regression Predicting Turnovers"))
```



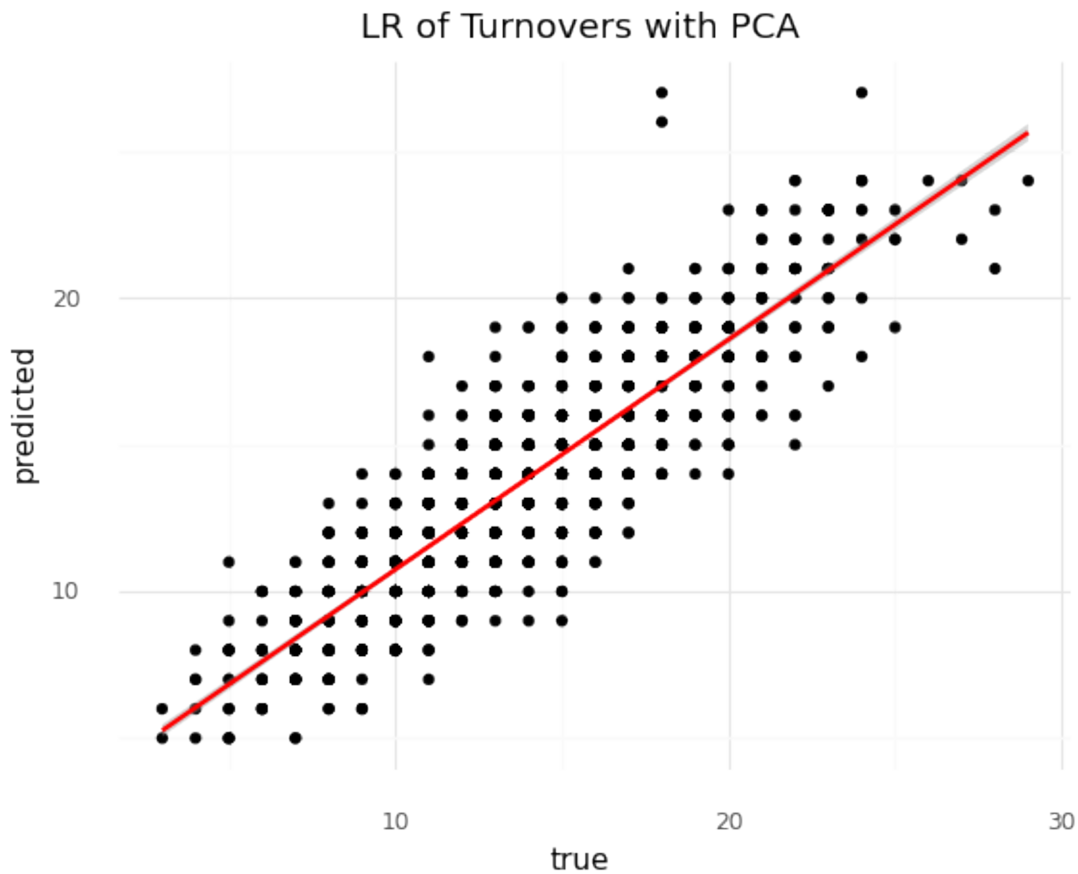

```
[ ]: <ggplot: (8734516658837)>
```

Figure 2.

Plot of the true vs predicted turnovers for our linear regression model using all continuous and standardized variables.

Linear Regression with Dimensionality Reduction

[ ]:
```
(ggplot(pca_true_vs_pred, aes(x = "true", y = "predicted")) + geom_point()+␣
↪theme_minimal() + geom_smooth(method = "lm", color = "red") + labs(title =␣
↪"LR of Turnovers with PCA"))
```



[ ]: <ggplot: (8734516018913)>

Figure 3.

Plot of the true vs predicted amount of turnovers with dimensionality redcution using principal component analysis and 14 principal components

Q1 Dicussion

2.0.1 When comparing a model using PCA on all the continuous variables for the team (not turnovers), what is the difference in linear regression accuracy between number of components when predicting how often a team will turnover in a game to retain enough PCs that explain 90% of the variance?

Many statistics were recorded across the NBA season games from 2014 to 2018. We used these continuous predictors, such as points and fouls, across games from this period in a linear regression model. This model predicted the number of turnovers given the continuous predictors. The model performed accurately on the train and test set with r2 scores of .80 and .81, respectively. This shows that our model is not overfitting since the model performed more accurately on the test set. Therefore, the r2 of our model explains above 80% of the variance in both seen and unseen data. However, using the many continuous statistics recorded in NBA games means our model is slightly inefficient and data-heavy. We can compare this model to a linear regression model that utilizes fewer statistics using principal component analysis. It does this by using some of the relationships between certain NBA variables. We found that 14 principal components, which are linear combinations of some of the original variables, were needed to explain 90% of the cumulative variance in the data. Using these 14 principal components, we created the second linear regression model to compare accuracies. The r2 of the PCA linear regression model for the train and test was .68 and .69, respectively. This means that our model with 14 pcs explains around 10% less variance in seen and unseen data than the original model. The mean square error using 14 components was also about four, while the original models were around 2 in both train and test sets. These accuracy results are also reflected in the linear regression model score between the two models, which had a difference of only .07. So while our linear regression model with all variables is more accurate and has less error in predictions of the turnovers in a game, the model using dimensionality reduction through 14 components was only somewhat less precise.

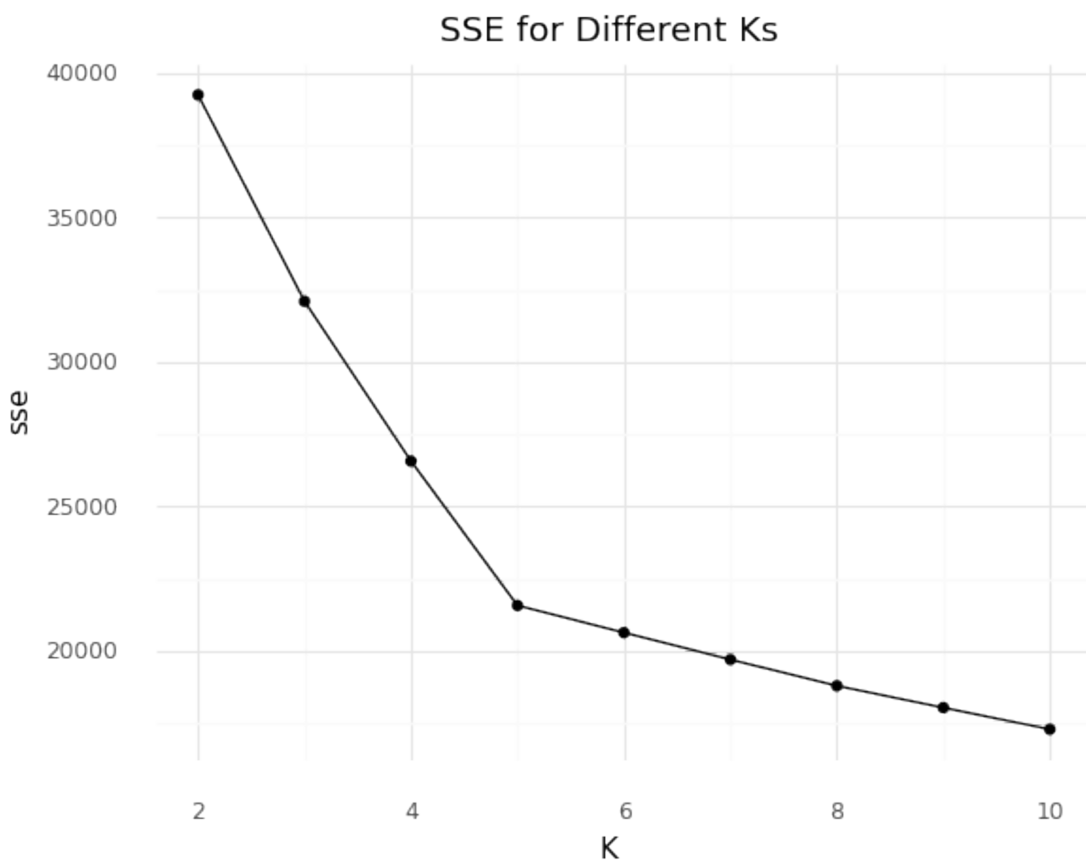
3 Q2

[ ]:
```
features = ['FieldGoals', 'Steals', 'TotalFouls', 'Blocks']
X = nba[features]

z = StandardScaler()

X[features] = z.fit_transform(X[features])

#model
km = KMeans(n_clusters = 5)
km.fit(X)

membership = km.predict(X)
X["cluster"] = membership

print(silhouette_score(X[features], membership))
```

```
0.17809731016970842
```

[ ]:
```
ks = [2,3,4,5,6,7,8,9,10]

sse = []
sils = []

for k in ks:
    km = KMeans(n_clusters = k)
```

```
    km.fit(X)

    sse.append(km.inertia_)
    sils.append(silhouette_score(X, km.predict(X)))

sse_df = pd.DataFrame({"K": ks,
                      "sse": sse,
                      "silhouette": sils})

(ggplot(sse_df, aes(x = "K", y = "sse")) + geom_point() +
geom_line() +
theme_minimal() +
labs(title = "SSE for Different Ks"))
```

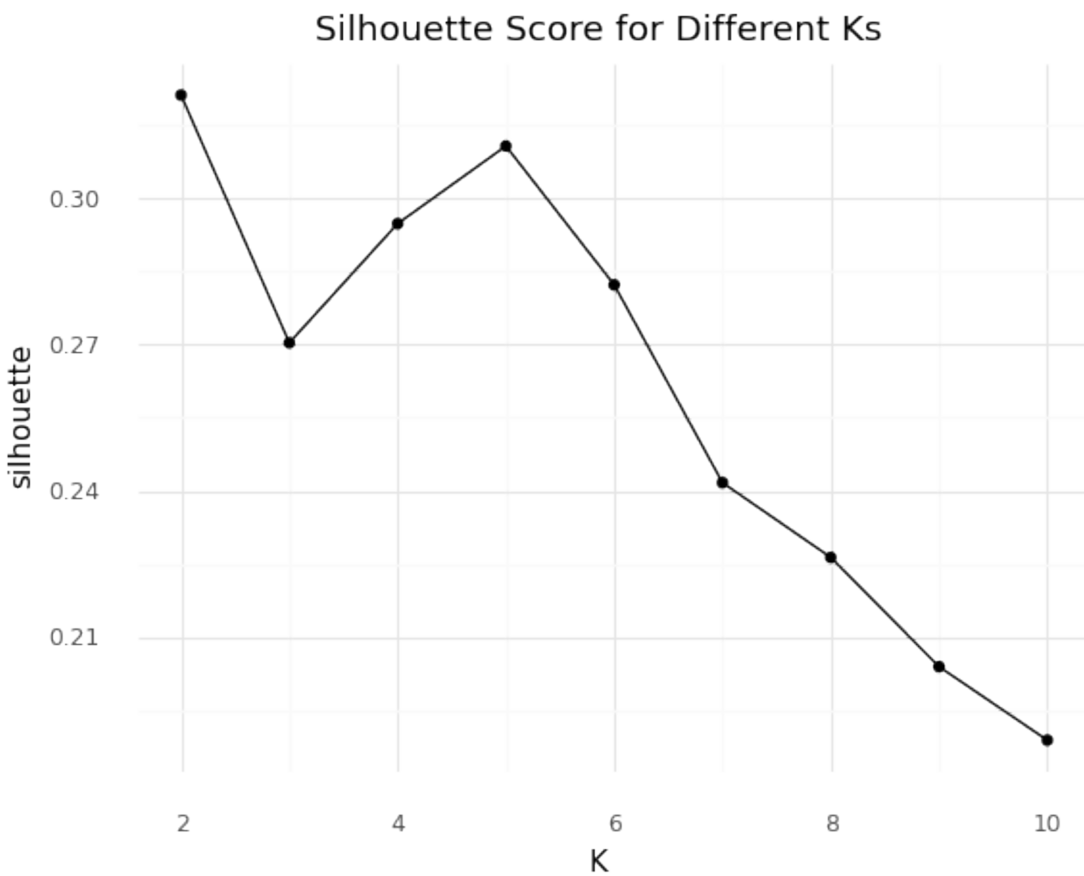


[ ]: <ggplot: (8734510851029)>

Figure 4.

Plot of the sum of squared error for increasing k cluster centers using Field Goals, Steals, Total Fouls and Blocks as predictors.

[ ]:
```
(ggplot(sse_df, aes(x = "K", y = "silhouette")) + geom_point() +
geom_line() +
theme_minimal() +
labs(title = "Silhouette Score for Different Ks"))
```

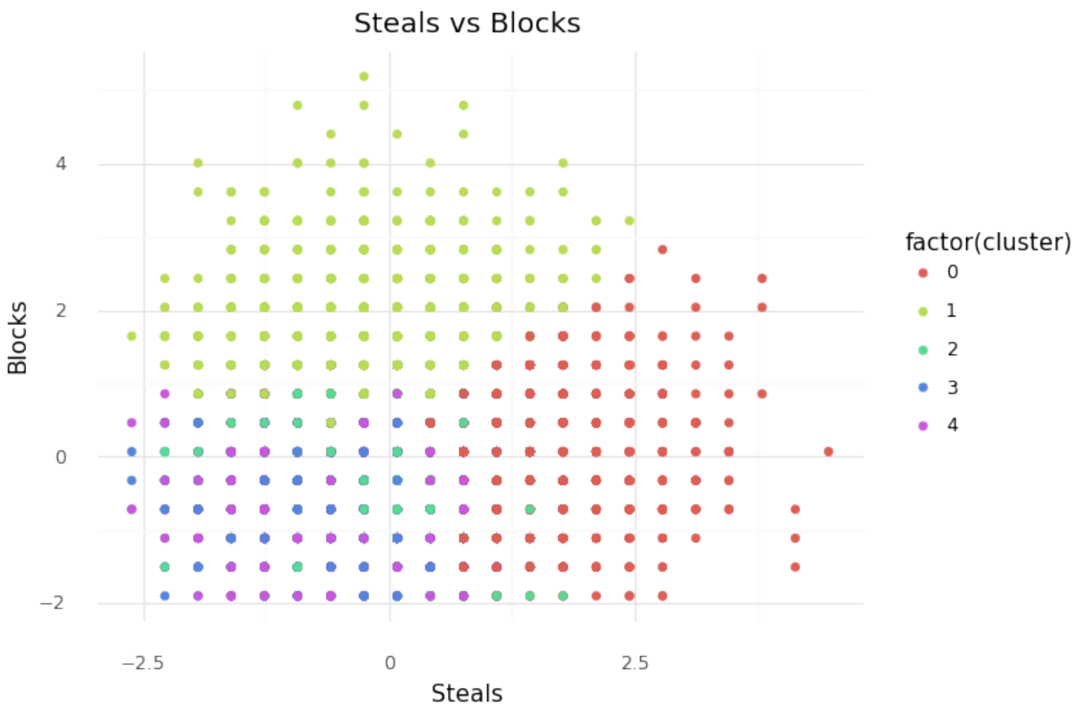


[ ]: <ggplot: (8734510823021)>

Figure 5.

Plot of the silhouette scores for increasing k cluster centers. Selected k = 5

[ ]:
```
(ggplot(X, aes(x = "Steals", y = "Blocks", color = "factor(cluster)")) +
 geom_point() + theme_minimal() + labs(title = "Steals vs Blocks"))
```

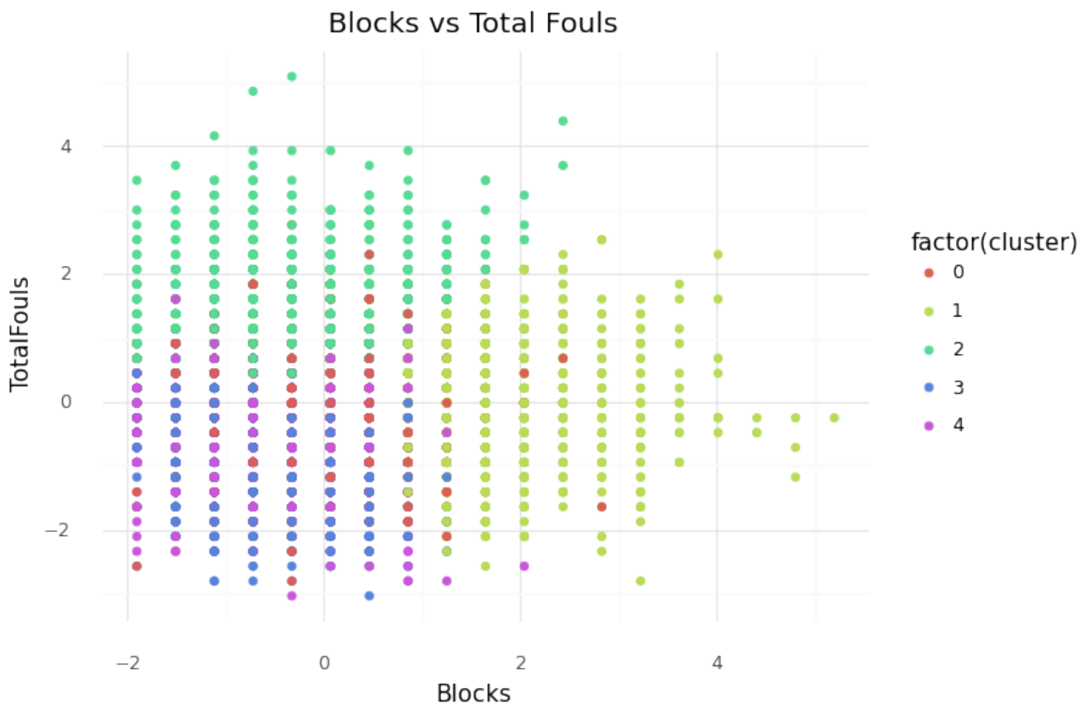


[ ]: <ggplot: (8734510761365)>

Figure 6.

Plot of the cluster assignment for steals vs blocks

```
[ ]: (ggplot(X, aes(x = "Blocks", y = "TotalFouls", color = "factor(cluster)")) +
     geom_point() + theme_minimal() + labs(title = "Blocks vs Total Fouls"))
```



[ ]: <ggplot: (8734510693285)>

Figure 7.

Plot of the cluster membership for block vs total fouls

3.1 Q2 Discussion

3.1.1 (Clustering) When considering field goals, steals, total fouls, and blocks what clusters emerge and what characterizes clusters based on these characteristics?

Plotting the relationship between these statistics recorded across games, clusters of the data have specific characteristics related to these stats. We can use KMeans to cluster these unique relationships in a certain amount of clusters. Initially, we need to test the sum of squared error for the model using different amounts of clusters. We can plot this error alongside the increasing k clusters, which helped identify 5 clusters as optimal. I also planned the silhouette scores of each increasing k, revealing 5 clusters as a point with a relatively high score above 0.3. This silhouette score represents the quality of the clusters we created in terms of how well similar data points are clustered with each other. Since our score is close to zero, some of our clusters poorly perform with the overlapping and dense data. However, our model still produced meaningful clusters with functional characteristics. After identifying and choosing 5 clusters for our model, we plotted the cluster assignments between steals vs. blocks, blocks vs. total fouls, and field goals vs. steals. Steals vs. blocks show how a game is played defensively by what method they are protecting the ball from the opposing team, gaining possession and ultimately points. Of the 5 clusters in Steals vs. blocks, the red cluster has games with more steals and some blocks. The green cluster shows games with more blocks and only some steals. This indicates that we can cluster games with more defense led through steals or blocks. In the blocks vs. total fouls plot, we can see how defensive play with blocks affects total fouls. In teal cluster 2, there are low blocks and high fouls. In the green cluster, there are high blocks and somewhat low fouls. The teal cluster can be seen as games that receive more fouls from other types of play than defensive plan. The green cluster shows games played more defensively through blocks have fewer fouls through offensive play. While our model was able to identify these clusters, there is a low accuracy because of the density and overlapping of the data. Using another method of clustering, we could further analyze these cluster relationships. However, KMeans was somewhat accurately able to reveal these unique cluster relationships for these predictors.

4 Export

[ ]:
```
# doesn't show this cells output when downloading PDF
!pip install gwpy &> /dev/null

# installing necessary files
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!sudo apt-get update
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic

# installing pypandoc
!pip install pypandoc

# connecting your google drive
from google.colab import drive
drive.mount('/content/drive')

# copying your file over. Change "Class6-Completed.ipynb" to whatever your file␣
↪is called (see top of notebook)
!cp "drive/My Drive/Colab Notebooks/NBA_Kearns_Analysis.ipynb" ./

# Again, replace "Class6-Completed.ipynb" to whatever your file is called (see␣
↪top of notebook)
!jupyter nbconvert --to PDF "NBA_Kearns_Analysis.ipynb"
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
pandoc is already the newest version (1.19.2.4~dfsg-1build4).
pandoc set to manually installed.
The following packages were automatically installed and are no longer required:
  libnvidia-common-460 nsight-compute-2020.2.0
```