FinalProject392JustinLewinski

May 17, 2023

[ ]:
```
import warnings
warnings.filterwarnings('ignore')
import warnings
warnings.filterwarnings('ignore')
import warnings
warnings.filterwarnings('ignore')
from plotnine import *
from sklearn.decomposition import PCA
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split # simple TT split cv
import numpy as np
import seaborn as sb
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
from plotnine import *
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.neighbors import KNeighborsClassifier


from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
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 accuracy_score, confusion_matrix,f1_score,␣
↪recall_score, precision_score, roc_auc_score
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪ConfusionMatrixDisplay
```

```python
from sklearn.model_selection import GridSearchCV

import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_blobs
import warnings
warnings.filterwarnings('ignore')
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor #
    ↪Decision Tree
import pandas as pd
import numpy as np
from plotnine import *

import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics
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.model_selection import LeaveOneOut #LOO cv

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

from sklearn.model_selection import GridSearchCV

from sklearn.datasets import make_blobs

%matplotlib inline

%matplotlib inline

from sklearn.preprocessing import LabelBinarizer

from sklearn.metrics import ConfusionMatrixDisplay

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

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import r2_score, mean_squared_error
```

```python
from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
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.preprocessing import LabelBinarizer

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import r2_score, mean_squared_error

from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
from sklearn.model_selection import LeaveOneOut #LOO cv

import warnings
warnings.filterwarnings('ignore')


import pandas as pd
import numpy as np


from sklearn.naive_bayes import GaussianNB, BernoulliNB, CategoricalNB #␣
↪Decision Tree
from sklearn.model_selection import train_test_split

from sklearn import metrics
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.model_selection import LeaveOneOut #LOO cv
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

import warnings
warnings.filterwarnings('ignore')

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, roc_auc_score, ConfusionMatrixDisplay
import pandas as pd
import numpy as np
```

```python
from plotnine import *

from sklearn.linear_model import LogisticRegression # Logistic Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, recall_score, precision_score, roc_auc_score
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
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


%matplotlib inline

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

from sklearn.preprocessing import StandardScaler #Z-score variables

from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture

from sklearn.metrics import silhouette_score

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
from plotnine import *
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
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.preprocessing import LabelBinarizer
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.pipeline import make_pipeline
```

```python
from sklearn.compose import make_column_transformer
import warnings
warnings.filterwarnings('ignore')

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

import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics
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.model_selection import LeaveOneOut #LOO cv

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

from sklearn.model_selection import GridSearchCV

from sklearn.datasets import make_blobs
from sklearn.tree import DecisionTreeClassifier
%matplotlib inline
```

#Variables ### attendance: how many people were at the game ### away_team: Name of away team ### away_team_errors: # of errors accumulated by the away team ### away_team_hits: # of hits accumulated by away team ### away_team_runs: # of runs accumulated by the away team ### date: day of game ### field_type: whether the field is on turf or grass ### game_type: whether the game takes place during the day or night ### home_team: Name of team playing at home ### home_team_errors: # of errors accumulated by the home team ### home_team_hits: # of hits accumulated by the home team ### home_team_runs: # of runs accumulated by the home team ### start_time: Time the game starts ### venue: name of stadium the game is played in. ### day_of_week: day the game is played ### temperature: degrees fahrenheit of the temperature during the game ### wind_speed: mph of wind during game ### wind_direction: which way the wind is traveling during the game ### sky: whether the sky is sunny, cloudy, overcast, or in a dome ### total_runs: sum of runs accumulated by both teams. ### game_hours_dec: amount of time the game lasted from start to finish ### season: whether the game is in the regular season or playoffs ### home_team_outcome: whether the home team won or lost

[ ]:
```python
data = pd.read_csv('/content/baseball_reference_2016_clean.csv')
data = data.dropna()
data.drop(columns=['Unnamed: 0'], inplace=True)
data.head()
```

```
[ ]:    attendance              away_team  away_team_errors  away_team_hits  \
     0     40030.0          New York Mets                 1               7
     1     21621.0  Philadelphia Phillies                 0               5
     2     12622.0        Minnesota Twins                 0               5
     3     18531.0   Washington Nationals                 0               8
     4     18572.0       Colorado Rockies                 1               8

        away_team_runs        date field_type   game_type             home_team  \
     0               3  2016-04-03   on grass  Night Game    Kansas City Royals
     1               2  2016-04-06   on grass  Night Game       Cincinnati Reds
     2               2  2016-04-06   on grass  Night Game     Baltimore Orioles
     3               3  2016-04-06   on grass  Night Game        Atlanta Braves
     4               4  2016-04-06   on grass    Day Game  Arizona Diamondbacks

        home_team_errors  …  temperature  wind_speed         wind_direction  \
     0                 0  …         74.0        14.0     from Right to Left
     1                 0  …         55.0        24.0     from Right to Left
     2                 0  …         48.0         7.0       out to Leftfield
     3                 1  …         65.0        10.0     from Right to Left
     4                 0  …         77.0         0.0   in unknown direction

             sky total_runs  game_hours_dec          season home_team_win  \
     0     Sunny          7        3.216667  regular season             1
     1  Overcast          5        2.383333  regular season             1
     2   Unknown          6        3.183333  regular season             1
     3    Cloudy          4        2.883333  regular season             0
     4   In Dome          7        2.650000  regular season             0

       home_team_loss  home_team_outcome
     0              0                Win
     1              0                Win
     2              0                Win
     3              1               Loss
     4              1               Loss

     [5 rows x 25 columns]
```

0.0.1 Question 1. (Supervised Model) When predicting the amount of runs scored, what implications does wind have on the outcome, and what are the most important factors of a high scoring game?

```
[ ]:
```

```
[ ]: dummy_variables = pd.get_dummies(data['game_type'], drop_first=True)


     data = pd.concat([data, dummy_variables], axis=1)
```

```
dummy_variables1 = pd.get_dummies(data['season'], drop_first=True)


data = pd.concat([data, dummy_variables1], axis=1)
dummy_variables2 = pd.get_dummies(data['field_type'], drop_first=True)


data = pd.concat([data, dummy_variables2], axis=1)

data.head()
```

[ ]:

```
   attendance              away_team  away_team_errors  away_team_hits  \
0     40030.0          New York Mets                 1               7
1     21621.0  Philadelphia Phillies                 0               5
2     12622.0        Minnesota Twins                 0               5
3     18531.0   Washington Nationals                 0               8
4     18572.0       Colorado Rockies                 1               8

   away_team_runs        date field_type   game_type            home_team  \
0               3  2016-04-03   on grass  Night Game   Kansas City Royals
1               2  2016-04-06   on grass  Night Game      Cincinnati Reds
2               2  2016-04-06   on grass  Night Game    Baltimore Orioles
3               3  2016-04-06   on grass  Night Game       Atlanta Braves
4               4  2016-04-06   on grass    Day Game  Arizona Diamondbacks

   home_team_errors  …        sky  total_runs game_hours_dec          season  \
0                 0  …      Sunny           7       3.216667  regular season
1                 0  …   Overcast           5       2.383333  regular season
2                 0  …    Unknown           6       3.183333  regular season
3                 1  …     Cloudy           4       2.883333  regular season
4                 0  …    In Dome           7       2.650000  regular season

  home_team_win  home_team_loss  home_team_outcome Night Game regular season  \
0             1               0                Win          1              1
1             1               0                Win          1              1
2             1               0                Win          1              1
3             0               1               Loss          1              1
4             0               1               Loss          0              1

   on turf
0        0
1        0
2        0
3        0
4        0

[5 rows x 28 columns]
```

[ ]:
```
data.columns
```

[ ]:
```
Index(['attendance', 'away_team', 'away_team_errors', 'away_team_hits',
       'away_team_runs', 'date', 'field_type', 'game_type', 'home_team',
       'home_team_errors', 'home_team_hits', 'home_team_runs', 'start_time',
       'venue', 'day_of_week', 'temperature', 'wind_speed', 'wind_direction',
       'sky', 'total_runs', 'game_hours_dec', 'season', 'home_team_win',
       'home_team_loss', 'home_team_outcome', 'Night Game', 'regular season',
       'on turf'],
      dtype='object')
```

[ ]:
```
cont = (['attendance', 'away_team_errors', 'away_team_hits',
        'home_team_errors', 'home_team_hits', 'temperature', 'wind_speed',␣
 ↪'game_hours_dec'])
lr1vars = (['attendance', 'away_team_errors', 'away_team_hits',
        'home_team_errors', 'home_team_hits','home_team_win',
        'temperature', 'wind_speed', 'game_hours_dec', 'Night Game',
        'regular season', 'on turf'])

X = data[lr1vars]
y = data[['total_runs']]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=392)
```

[ ]:
```
z = StandardScaler()
z.fit(X_train[cont])
X_train[cont] = z.transform(X_train[cont])
X_test[cont] = z.transform(X_test[cont])

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

[ ]:
```
LinearRegression()
```

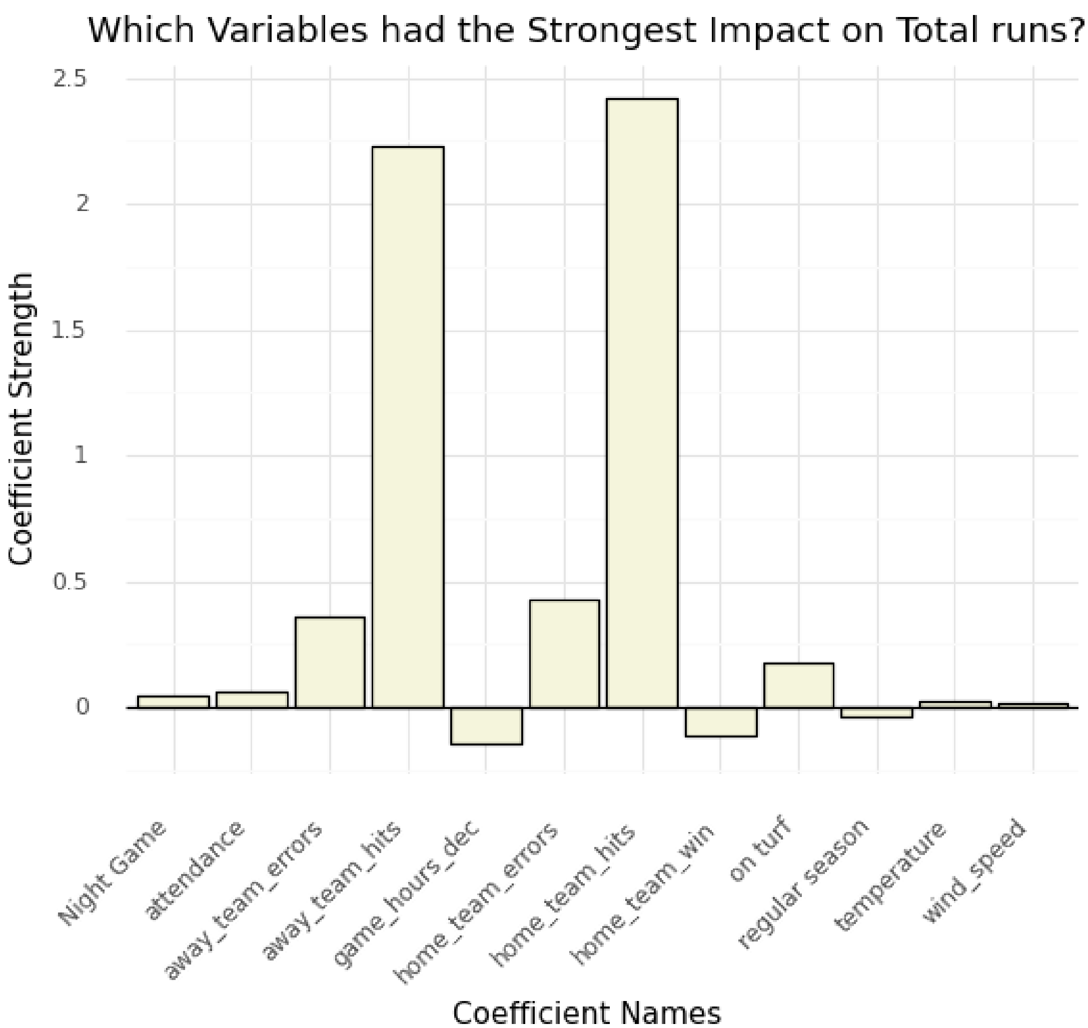
[ ]:
```
coef = pd.DataFrame({"Coef": lr.coef_[0], "Names": lr1vars})
coef = coef.append({"Coef": lr.intercept_[0], "Names": "intercept"},
ignore_index = True)
coef = coef.drop(index = 12)
coef
```

[ ]:

	Coef	Names
0	0.063829	attendance
1	0.360105	away_team_errors
2	2.231224	away_team_hits
3	0.431349	home_team_errors
4	2.422445	home_team_hits
5	-0.115721	home_team_win

```
6   0.026201       temperature
7   0.016231        wind_speed
8  -0.141845    game_hours_dec
9   0.049878        Night Game
10 -0.035693    regular season
11  0.178710           on turf
```

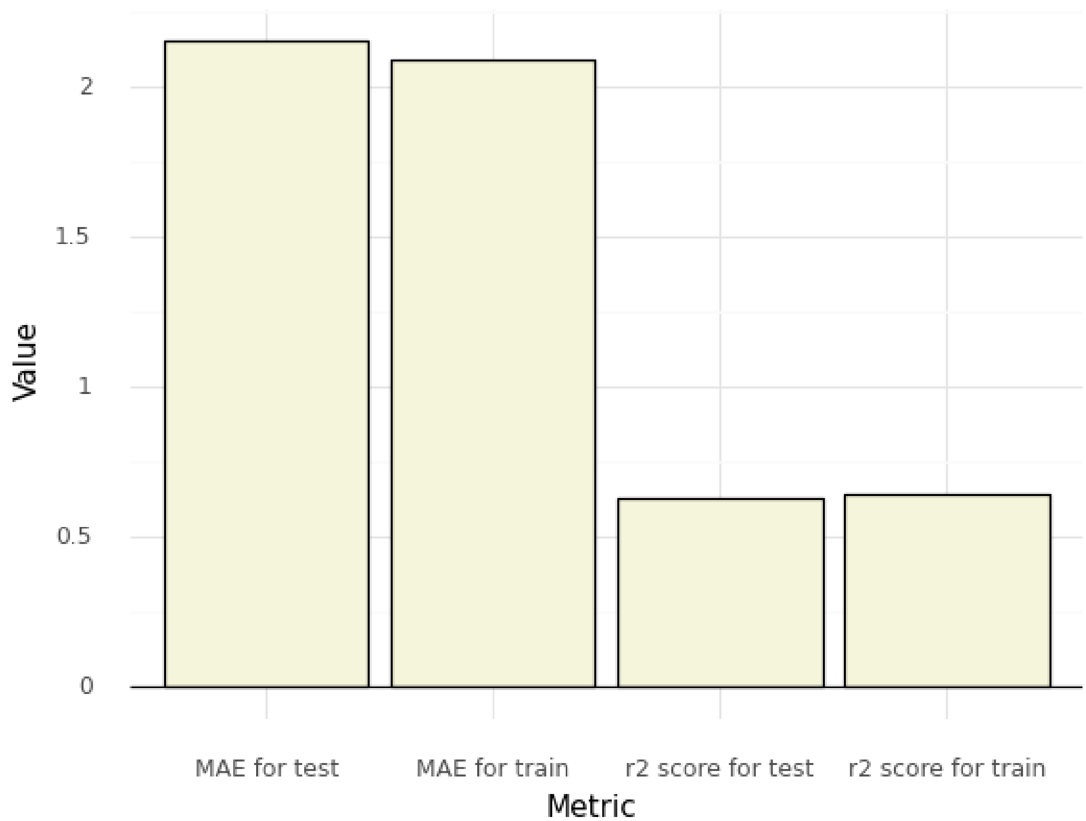
[ ]:
```
(ggplot(coef, aes(x='Names', y='Coef'))
    + geom_bar(stat='identity', color = 'black', fill = 'beige')
    + labs(x='Coefficient Names', y='Coefficient Strength', title='Which␣
 ↪Variables had the Strongest Impact on Total runs?')
    + theme_minimal()
    + theme(axis_text_x=element_text(angle=45, ha='right'))
    + geom_hline(yintercept = 0))
```



[ ]: <ggplot: (8757948697594)>

[ ]:
```
y_pred = lr.predict(X_test)
lrMaeTrain = mean_absolute_error(y_train, lr.predict(X_train))
lrMaeTest = mean_absolute_error(y_test, y_pred)
print('MAE for train: ', lrMaeTrain)
print('MAE for test: ',lrMaeTest)
lrR2Train = r2_score(y_train, lr.predict(X_train))
lrR2Test = r2_score(y_test, lr.predict(X_test))
print('r2 score for train: ',lrR2Train)
print('r2 score for test: ',lrR2Test)

values = {
    'Metric': ['MAE for train', 'MAE for test', 'r2 score for train', 'r2 score␣
 ↪for test'],
    'Value': [lrMaeTrain, lrMaeTest, lrR2Train, lrR2Test]
}


dfq1error = pd.DataFrame(values)

(ggplot(dfq1error, aes(x = 'Metric', y = 'Value'))
 + geom_bar(stat = 'identity', color = 'black', fill = 'beige')
 + theme_minimal()
 + geom_hline(yintercept = 0))
```

```
MAE for train:  2.0915331060027404
MAE for test:  2.154432060283022
r2 score for train:  0.6404691912866918
r2 score for test:  0.6307749957695851
```



[ ]: <ggplot: (8757948775740)>

0.1 Based on the model's performance (displayed by the metric bar graph) It is apparent that one most likely needs more telling statistics to accurately predict the amount of runs scored in a game. Perhaps individual pitching and hitting stats of the players. However, in the model constructed, wind_speed has an almost non-existent effect on total runs scored in a game, which goes against my initial hypothesis. The most important factors that contribute to a high scoring game are hits and errors by both the home team and away team. This makes sense to those who follow baseball, and there does not seem to be any surprising variable impacts that were unexpected.

0.2 Question 2: Which stadium had the best home team advantage? Which team won the most away games?

[ ]:
```
q2 = ['venue', 'home_team', 'home_team_win']
data[q2]
q2_sorted = data[q2].groupby('venue').sum().reset_index().
 ↪sort_values('home_team_win', ascending=False)
```

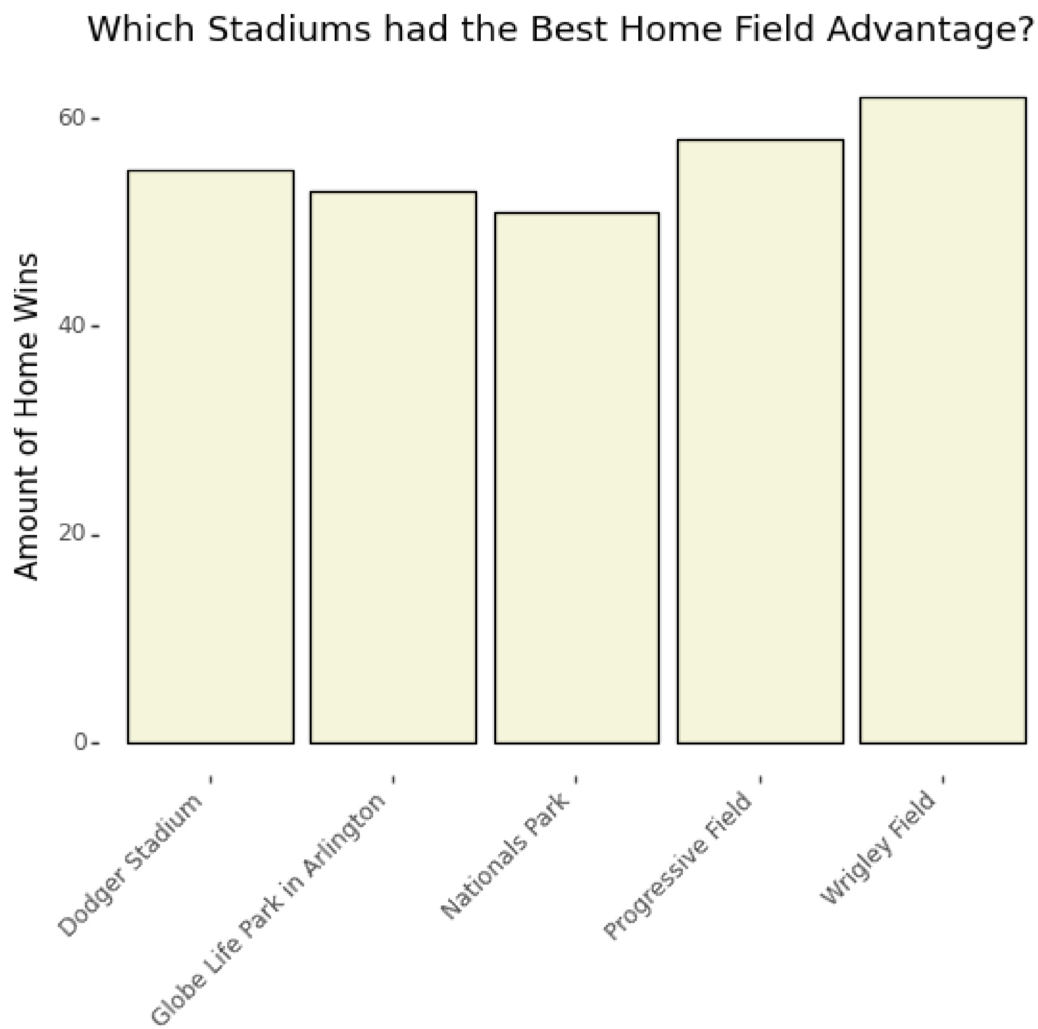
```
top_5 = q2_sorted.head(5)
top_5
```

[ ]:

```
                          venue  home_team_win
29                Wrigley Field             62
22            Progressive Field             58
8                Dodger Stadium             55
11  Globe Life Park in Arlington             53
17               Nationals Park             51
```

[ ]:

```
plot = (ggplot(top_5, aes(x='venue', y='home_team_win')) +
       geom_bar(stat='identity', color = 'black', fill = 'beige') +␣
 ↪theme(axis_text_x=element_text(angle=45, ha='right'))
       + theme(panel_background=element_rect(fill='white')) + labs(x = ' ', y =␣
 ↪'Amount of Home Wins', title = 'Which Stadiums had the Best Home Field␣
 ↪Advantage?'))



plot
```

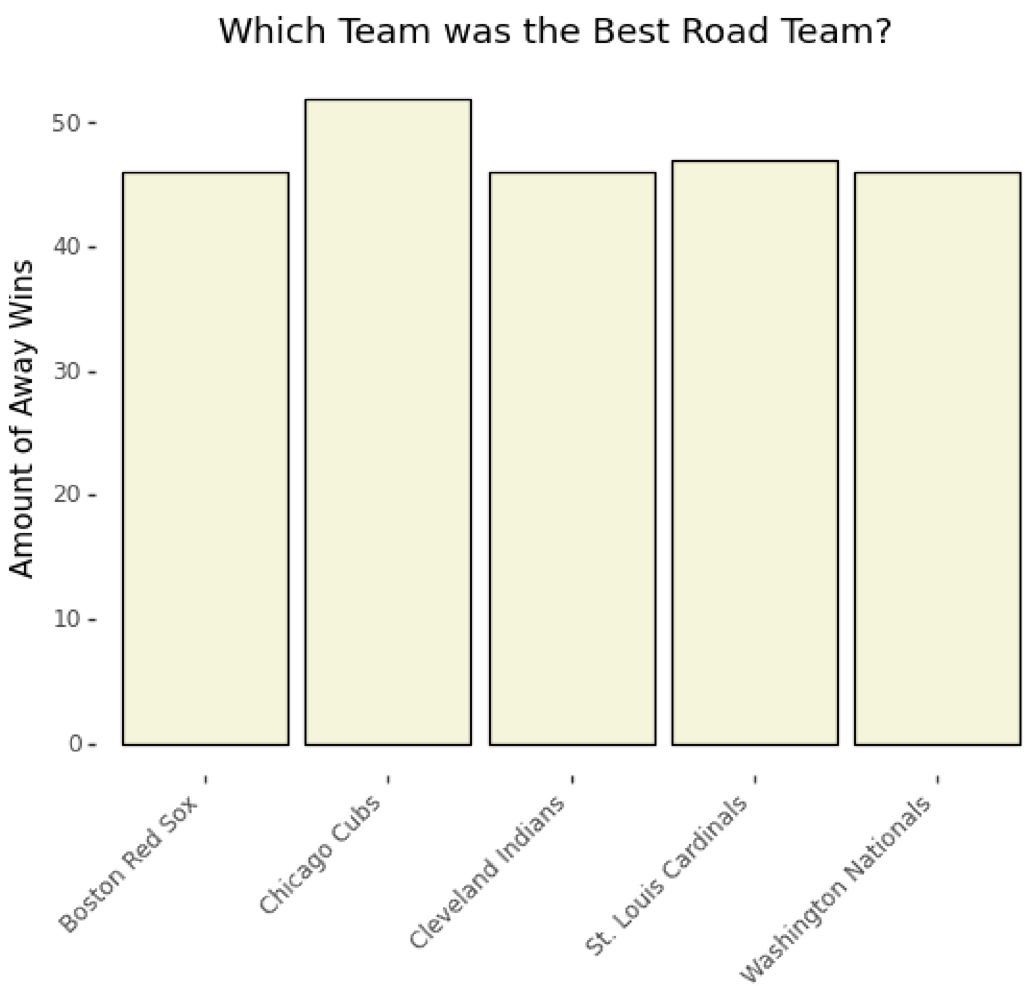


[ ]: <ggplot: (8757944290437)>

```
[ ]: q2feats2 = ['away_team', 'home_team_loss']

     data[q2feats2]

     q2_sorted2 = data[q2feats2].groupby('away_team').sum().reset_index().
      ↪sort_values('home_team_loss', ascending=False)

     top_5_away = q2_sorted2.head(5)

     top_5_away
```

```
[ ]:                away_team  home_team_loss
    4            Chicago Cubs              52
    25    St. Louis Cardinals              47
    29  Washington Nationals               46
    7       Cleveland Indians              46
    3          Boston Red Sox              46
```

```
[ ]: (ggplot(top_5_away, aes(x='away_team', y='home_team_loss')) +
            geom_bar(stat='identity', color = 'black', fill = 'beige') +␣
      ↪theme(axis_text_x=element_text(angle=45, ha='right'))
            + theme(panel_background=element_rect(fill='white')) + labs(x = ' ', y =␣
      ↪'Amount of Away Wins', title = 'Which Team was the Best Road Team?'))
```




```
[ ]: <ggplot: (8757944291187)>
```

0.3 **When one looks at both the home stadium bar chart as well as the best road team bar chart, It becomes apparent that the field with the best homefield advantage in 2016 was Wrigley field. Wrigley field was host to the most home wins of any stadium, as their team, the Chicago Cubs, had an incredible year. The best road team was ALSO the Chicago Cubs, further displaying why they went on to win the world series in 2016.**

0.4 **Question 3. (Dimensionality Reduction) When comparing a model using PCA on all the continuous variables in the dataset and retaining enough PCs to keep 90% of the variance, to a model using all the continuous variables, how much of a difference is there in accuracy when predicting home_team_win?**

[ ]:
```
X = data[cont]
y = data['home_team_win']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=392)

z = StandardScaler()
X_train[cont] = z.fit_transform(X_train)
X_test[cont] = z.transform(X_test)

logit = LogisticRegression()
logit.fit(X_train, y_train)

coef = pd.DataFrame({"Coef": logit.coef_[0], "Names": cont})
coef = coef.append({"Coef": logit.intercept_[0], "Names":␣
↪"intercept"},ignore_index = True)
coef["Odds"] = np.exp(coef["Coef"])
coef
```

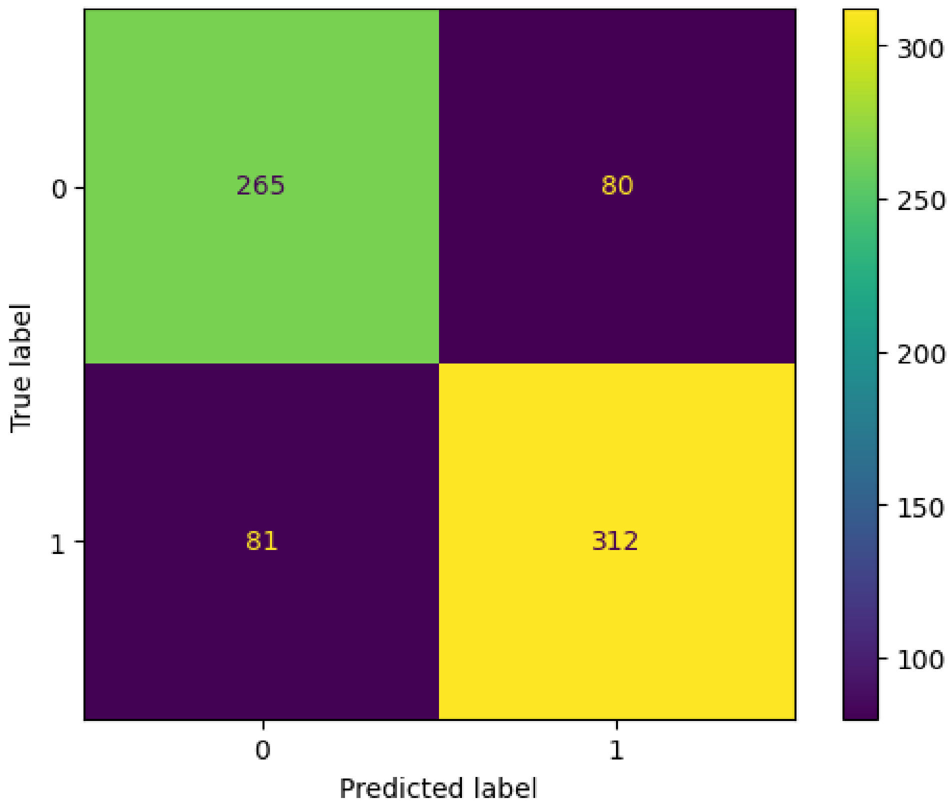
[ ]:

	Coef	Names	Odds
0	0.060629	attendance	1.062505
1	0.149852	away_team_errors	1.161662
2	-1.549121	away_team_hits	0.212435
3	-0.274306	home_team_errors	0.760099
4	1.614678	home_team_hits	5.026267
5	0.010414	temperature	1.010469
6	0.104771	wind_speed	1.110456
7	-0.336177	game_hours_dec	0.714496
8	0.193945	intercept	1.214030

[ ]:
```
predictedVals = logit.predict(X_test) #predict
predictedProbs = logit.predict_proba(X_test)

print("Accuracy: ", accuracy_score(y_test, predictedVals))
print("F1 Score: ", f1_score(y_test, predictedVals))
```

```python
print("Recall: ", recall_score(y_test, predictedVals))
print("Precision: ", precision_score(y_test, predictedVals))

matrix = confusion_matrix(y_test, predictedVals)
disp = ConfusionMatrixDisplay(matrix)
disp.plot()
plt.show()
```

```
Accuracy:  0.7818428184281843
F1 Score:  0.794904458598726
Recall:  0.7938931297709924
Precision:  0.7959183673469388
```

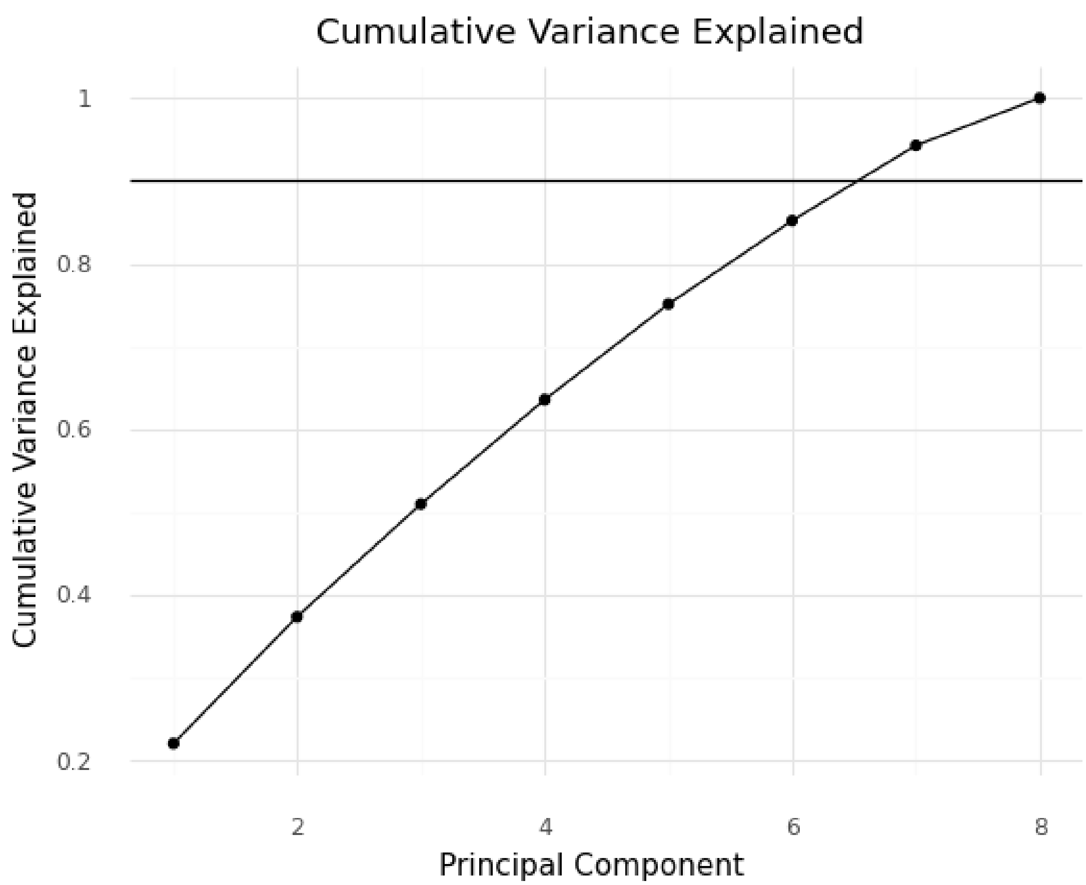


[ ]:
```python
pd.DataFrame({'Accuracy':  0.7818428184281843,
'F1 Score' :  0.794904458598726,
'Recall' :  0.7938931297709924,
'Precision' :  0.7959183673469388}, index = [0])
```

[ ]:

	Accuracy	F1 Score	Recall	Precision
0	0.781843	0.794904	0.793893	0.795918

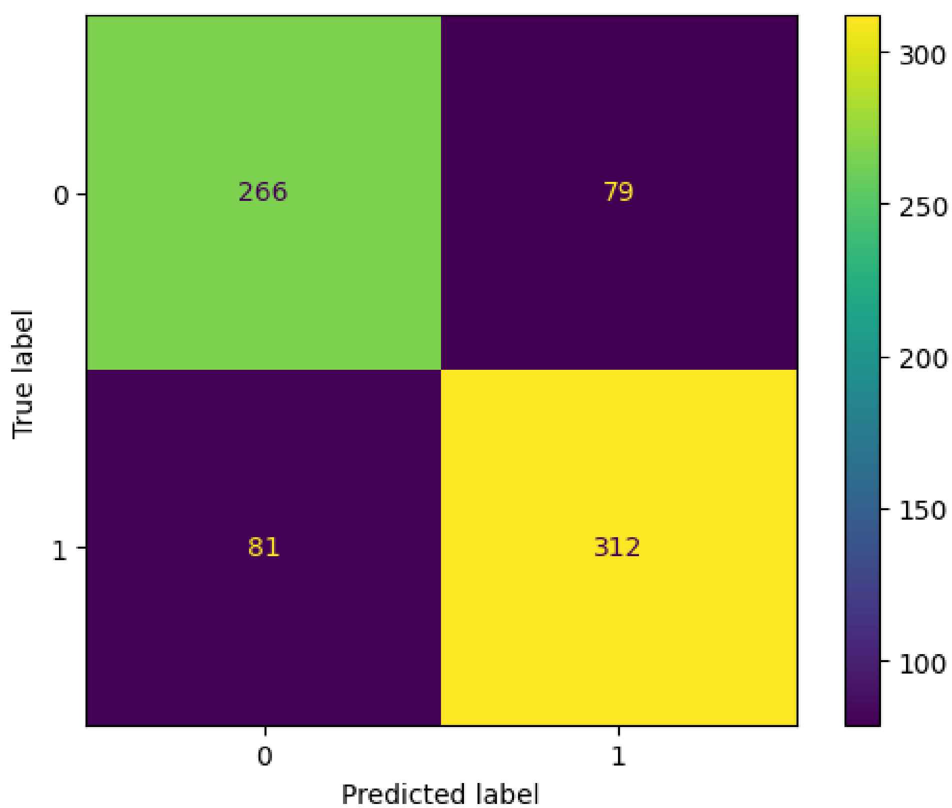
[ ]:
```
pca = PCA()
pca.fit(X_train)
cumulative_variance_df = pd.DataFrame({'Principal Component': range(1, len(pca.
↪explained_variance_ratio_) + 1),
                                       'Cumulative Variance Explained': np.
↪cumsum(pca.explained_variance_ratio_)})

(ggplot(cumulative_variance_df, aes(x='Principal Component', y='Cumulative␣
↪Variance Explained')) +
 geom_point() +
 geom_line() +
 labs(x='Principal Component', y='Cumulative Variance Explained',␣
↪title='Cumulative Variance Explained') +
 geom_hline(yintercept = 0.9) + theme_minimal())
```



[ ]:
```
<ggplot: (8757944174550)>
```

```
[ ]: pca = PCA(n_components=7)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
pcalogit = LogisticRegression()
pcalogit.fit(X_train_pca, y_train)
y_pred = pcalogit.predict(X_test_pca)
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("F1 Score: ", f1_score(y_test, y_pred))
print("Recall: ", recall_score(y_test, y_pred))
print("Precision: ", precision_score(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
cm_display = ConfusionMatrixDisplay(cm)
cm_display.plot()
plt.show()
```

```
Accuracy:  0.7831978319783198
F1 Score:  0.7959183673469388
Recall:  0.7938931297709924
Precision:  0.7979539641943734
```



[ ]:
```
pd.DataFrame({'Accuracy' :  0.7831978319783198,
'F1 Score' :  0.7959183673469388,
'Recall' :  0.7938931297709924,
'Precision' :  0.7979539641943734}, index = [0])
```

[ ]:

	Accuracy	F1 Score	Recall	Precision
0	0.783198	0.795918	0.793893	0.797954

0.5 In an EXTREMELY surprising development, the PCA model with only 7 principle components performed better than the original model with the entire training set in it. This shocked me, as this is not typically the case. However, upon looking at the cumulative variance plot, it makes sense how something like this *could have* happened. Because there was not a certain group of important PCs, the difference between 90% explained variance and 100% is not that large. Thus the model's efficiency and accuracy is not as affected in the particular instance.

0.6 Question 4. When looking at the total runs scored per game, Which stadium hosted the highest scoring games on average? Which stadium had the least deviation in their total outcomes?

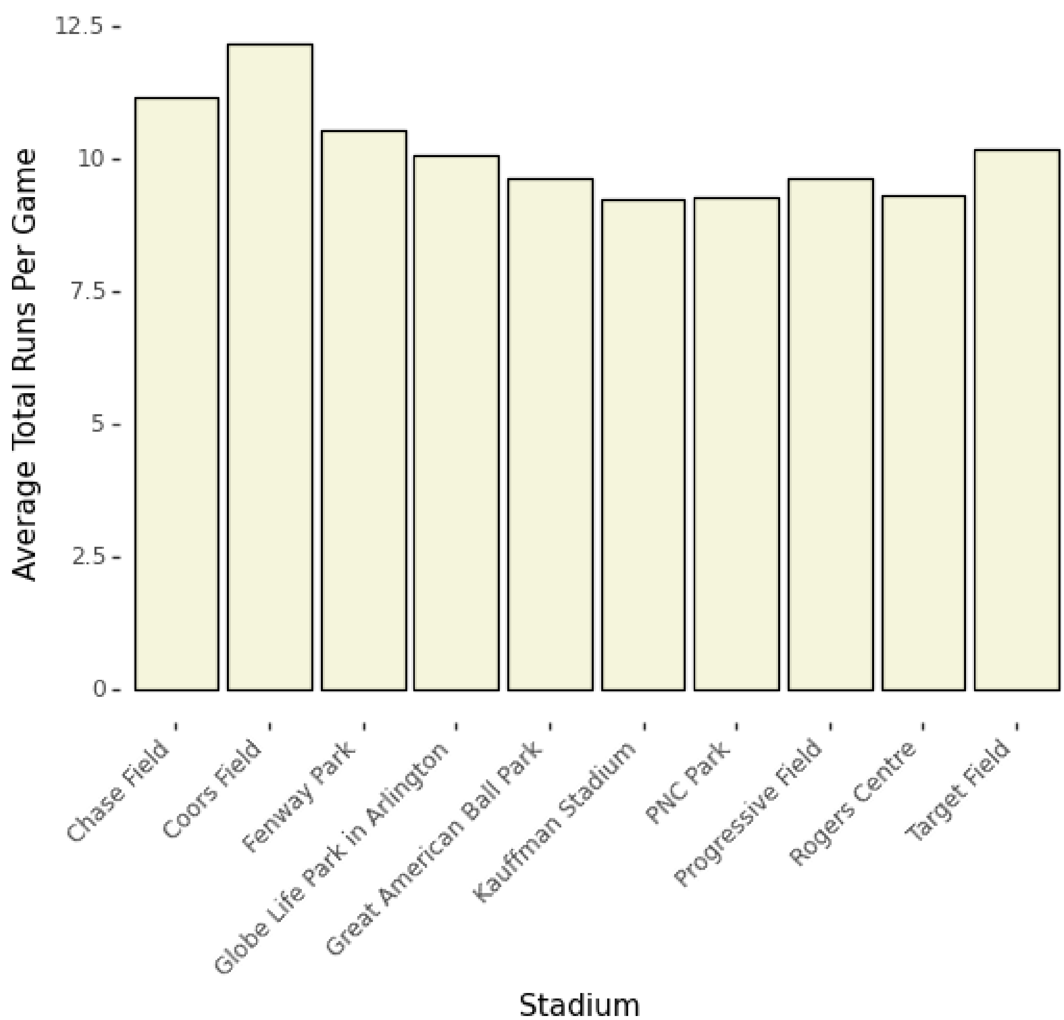
[ ]:
```
q4feats = ['venue', 'total_runs']
q4df = data[q4feats]

mean_total_runs_by_stadium = q4df.groupby('venue')['total_runs'].mean().
 ↪reset_index()
mtrbs = mean_total_runs_by_stadium.rename(columns={'total_runs':␣
 ↪'average_runs'})
mtrbs = mtrbs.sort_values(by='average_runs', ascending= False)
mtrbs = mtrbs.reset_index(drop=True)
mtrbs.head(24)
```

[ ]:

	venue	average_runs
0	Coors Field	12.160494
1	Chase Field	11.160494
2	Fenway Park	10.536585
3	Target Field	10.160494
4	Globe Life Park in Arlington	10.060241
5	Great American Ball Park	9.641975
6	Progressive Field	9.640449
7	Rogers Centre	9.290698
8	PNC Park	9.275000
9	Kauffman Stadium	9.234568
10	Comerica Park	9.225000
11	Turner Field	9.150000
12	Petco Park	9.049383
13	Safeco Field	8.827160

```
14              Busch Stadium III      8.827160
15   Oriole Park at Camden Yards      8.790123
16             Yankee Stadium III      8.679012
17                    Miller Park      8.543210
18       Angel Stadium of Anaheim      8.493827
19                     AT&T Park      8.421687
20             U.S. Cellular Field      8.262500
21                 Nationals Park      8.250000
22                Tropicana Field      8.049383
23              Citizens Bank Park      7.925926
```

[ ]:
```
(ggplot(mtrbs.head(10), aes(x='venue', y='average_runs'))
+ geom_bar(stat='identity', color = 'black', fill = 'beige')
+ theme(axis_text_x=element_text(angle=45, ha='right'))
+ theme(panel_background=element_rect(fill='white'))
+ labs(y = 'Average Total Runs Per Game', x = 'Stadium'))
```

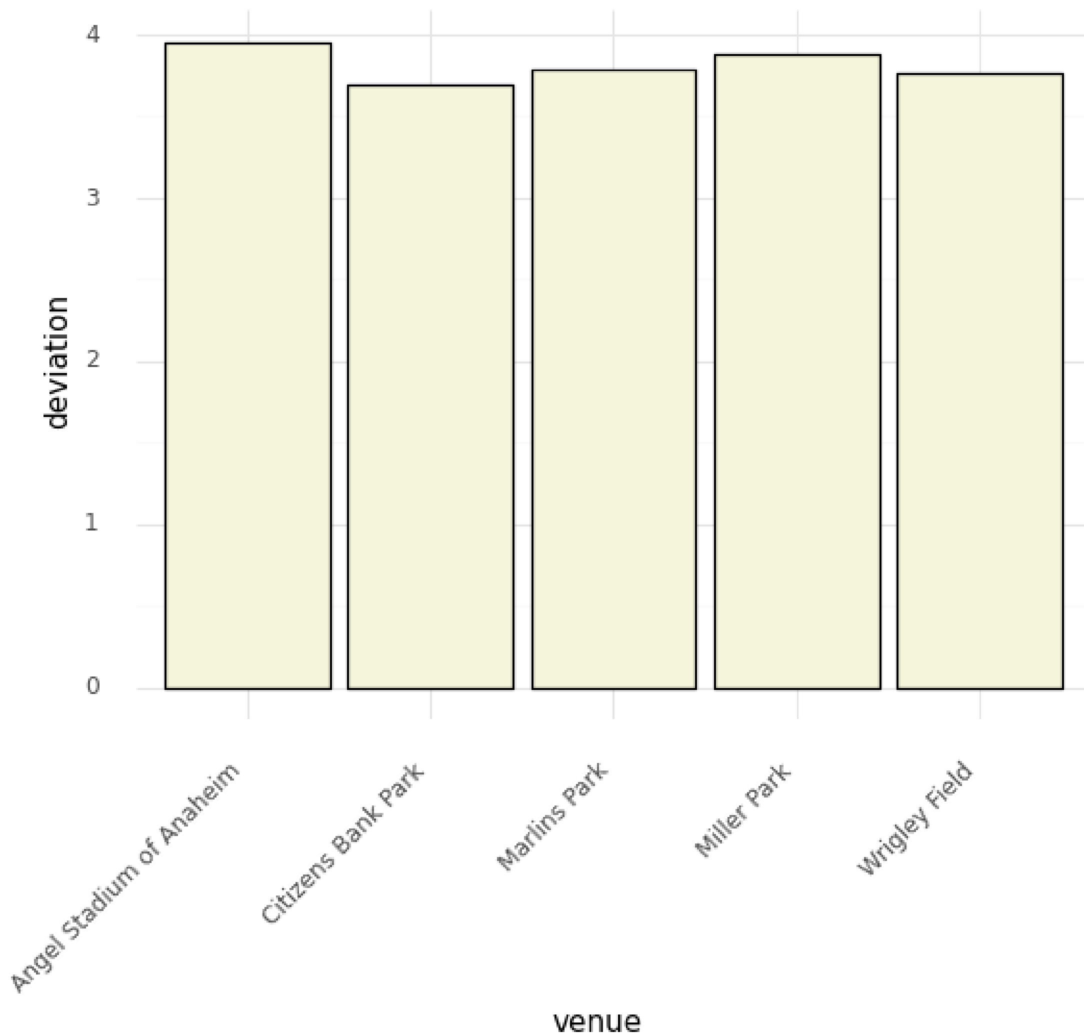


[ ]: <ggplot: (8757944174319)>

[ ]:
```
dps = q4df.groupby('venue')['total_runs'].std().reset_index()
dps = dps.rename(columns={'total_runs': 'deviation'})
dps = dps.sort_values(by='deviation', ascending= True)

dps.head()
```

[ ]:
```
                      venue  deviation
5         Citizens Bank Park   3.697221
29             Wrigley Field   3.764723
14              Marlins Park   3.785047
15               Miller Park   3.879592
1   Angel Stadium of Anaheim   3.956398
```

[ ]:
```
(ggplot(dps.head(5), aes(x = 'venue', y = 'deviation'))
+ geom_bar(stat = 'identity', color = 'black', fill = 'beige')
+ theme_minimal()
+ theme(axis_text_x=element_text(angle=45, ha='right')))
```

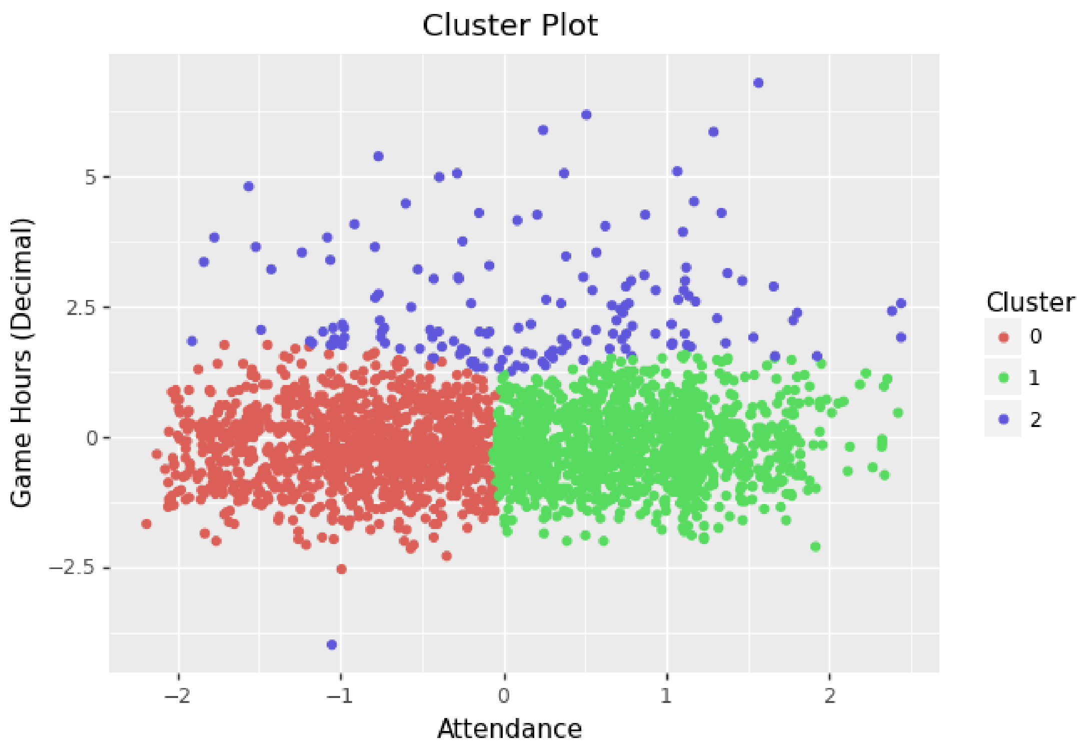



```
[ ]: <ggplot: (8757946487316)>
```

0.7 The Stadium that hosted the highest scoring games on average was Coors Field, with an average runs per game of ~12.16. The Stadium that hosts games with the least deviation in total_runs, at 3.7, is Citizens Bank Park, Home of the Philadelphia Phillies. With an Average total runs per game count of 7.926, You can expect the total run count of each game to be between 4 and 11 runs.

0.8 Question 5. (Clustering) When looking at attendance and length of the game, what clusters appear using a GMM?

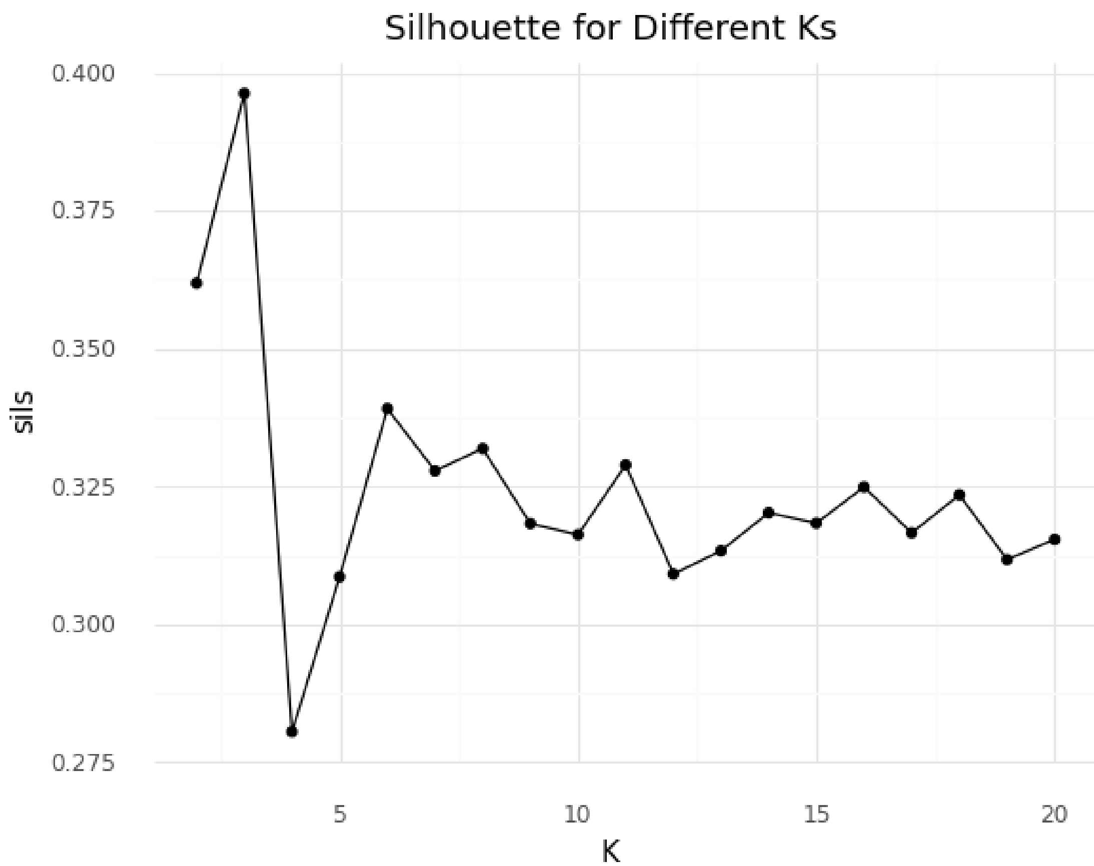
[ ]:
```
q5feats = ['attendance', 'game_hours_dec']
q5df = data[q5feats]
q5df.head()

X = q5df

z = StandardScaler()
X = z.fit_transform(X)

gmm = GaussianMixture(n_components = 3)
gmm.fit(X)

labels = gmm.predict(X)


clustered_data = pd.DataFrame({'attendance': X[:, 0], 'game_hours_dec': X[:,␣
↪1], 'cluster': labels})

ggplot(clustered_data, aes(x='attendance', y='game_hours_dec',␣
↪color='factor(cluster)')) + \
       geom_point() + \
       labs(title='Cluster Plot', x='Attendance', y='Game Hours (Decimal)',␣
↪color='Cluster')
```



[ ]: <ggplot: (8757944021398)>

[ ]:
```
X = q5df

z = StandardScaler()
X = z.fit_transform(X)

ks = [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

sils = []

for k in ks:
  gmm = GaussianMixture(n_components = k)
  gmm.fit(X)

  sils.append(silhouette_score(X, gmm.predict(X)))

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

(ggplot(sil_df, aes(x = "K", y = "sils")) + geom_point() +
geom_line() +
theme_minimal() +
```

```
labs(title = "Silhouette for Different Ks"))
```



[ ]: <ggplot: (8757944225717)>

0.9 When assessing the slihouette scores for the varying amounts of k, it is apparent that 3 clusters is the most optimal, and also the silhouette scores are low in general for this specific scatterplot. I would classify the first clusters games as "Short and small" to mean that the games in this cluster did not last long and also did not have high attendance. The Second cluster I would classify as "Short and Loud", as Even though the games did not last long, there was high attendace. The third cluster I would classify as "The Epics" due to all of their longer game times, although this cluster varies in attendance.

0.10 Question 6. (Supervised Model) Looking at the coefficients, do environmental variables (weather, day/night) have an impact on attendance? Does whether the game is a regular season game or playoff game matter?

[ ]:
```
q6df = data
```

[ ]:
```
q6df.columns
```

[ ]:
```
Index(['attendance', 'away_team', 'away_team_errors', 'away_team_hits',
       'away_team_runs', 'date', 'field_type', 'game_type', 'home_team',
       'home_team_errors', 'home_team_hits', 'home_team_runs', 'start_time',
       'venue', 'day_of_week', 'temperature', 'wind_speed', 'wind_direction',
       'sky', 'total_runs', 'game_hours_dec', 'season', 'home_team_win',
       'home_team_loss', 'home_team_outcome', 'Night Game', 'regular season',
       'on turf'],
      dtype='object')
```

[ ]:
```
dummyq6 = pd.get_dummies(q6df, columns=['sky', 'wind_direction', 'day_of_week',␣
↪'venue'])

dummyq6.columns
```

[ ]:
```
Index(['attendance', 'away_team', 'away_team_errors', 'away_team_hits',
       'away_team_runs', 'date', 'field_type', 'game_type', 'home_team',
       'home_team_errors', 'home_team_hits', 'home_team_runs', 'start_time',
       'temperature', 'wind_speed', 'total_runs', 'game_hours_dec', 'season',
       'home_team_win', 'home_team_loss', 'home_team_outcome', 'Night Game',
       'regular season', 'on turf', 'sky_Cloudy', 'sky_Drizzle', 'sky_In Dome',
       'sky_Night', 'sky_Overcast', 'sky_Rain', 'sky_Sunny', 'sky_Unknown',
       'wind_direction_ from Left to Right',
       'wind_direction_ from Right to Left',
       'wind_direction_ in from Centerfield',
       'wind_direction_ in from Leftfield',
       'wind_direction_ in from Rightfield',
       'wind_direction_ in unknown direction',
       'wind_direction_ out to Centerfield',
       'wind_direction_ out to Leftfield', 'wind_direction_ out to Rightfield',
```

```
        'day_of_week_Friday', 'day_of_week_Monday', 'day_of_week_Saturday',
        'day_of_week_Sunday', 'day_of_week_Thursday', 'day_of_week_Tuesday',
        'day_of_week_Wednesday', 'venue_AT&T Park',
        'venue_Angel Stadium of Anaheim', 'venue_Busch Stadium III',
        'venue_Chase Field', 'venue_Citi Field', 'venue_Citizens Bank Park',
        'venue_Comerica Park', 'venue_Coors Field', 'venue_Dodger Stadium',
        'venue_Fenway Park', 'venue_Fort Bragg Park',
        'venue_Globe Life Park in Arlington', 'venue_Great American Ball Park',
        'venue_Kauffman Stadium', 'venue_Marlins Park', 'venue_Miller Park',
        'venue_Minute Maid Park', 'venue_Nationals Park',
        'venue_Oakland-Alameda County Coliseum',
        'venue_Oriole Park at Camden Yards', 'venue_PNC Park',
        'venue_Petco Park', 'venue_Progressive Field', 'venue_Rogers Centre',
        'venue_Safeco Field', 'venue_Target Field', 'venue_Tropicana Field',
        'venue_Turner Field', 'venue_U.S. Cellular Field',
        'venue_Wrigley Field', 'venue_Yankee Stadium III'],
       dtype='object')
```

[ ]:
```
q6preds = ['wind_speed','temperature','Night Game', 'regular season',
       'on turf', 'day_of_week_Monday', 'day_of_week_Saturday',
       'day_of_week_Sunday', 'day_of_week_Thursday', 'day_of_week_Tuesday',
       'day_of_week_Wednesday', 'sky_Drizzle', 'sky_In Dome', 'sky_Night',␣
 ↪'sky_Overcast', 'sky_Rain',
       'sky_Sunny', 'sky_Unknown', 'wind_direction_ from Right to Left',
       'wind_direction_ in from Centerfield',
       'wind_direction_ in from Leftfield',
       'wind_direction_ in from Rightfield',
       'wind_direction_ in unknown direction', 'venue_AT&T Park',
       'venue_Angel Stadium of Anaheim', 'venue_Busch Stadium III',
       'venue_Chase Field', 'venue_Citi Field', 'venue_Citizens Bank Park',
       'venue_Comerica Park', 'venue_Coors Field', 'venue_Dodger Stadium',
       'venue_Fenway Park','venue_Globe Life Park in Arlington', 'venue_Great␣
 ↪American Ball Park',
       'venue_Kauffman Stadium', 'venue_Marlins Park', 'venue_Miller Park',
       'venue_Minute Maid Park', 'venue_Nationals Park',
       'venue_Oakland-Alameda County Coliseum',
       'venue_Oriole Park at Camden Yards', 'venue_PNC Park',
       'venue_Petco Park']
cont = ['wind_speed','temperature']
X = dummyq6[q6preds]
y = dummyq6['attendance']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=392)

z = StandardScaler()
z.fit(X_train[cont])
```

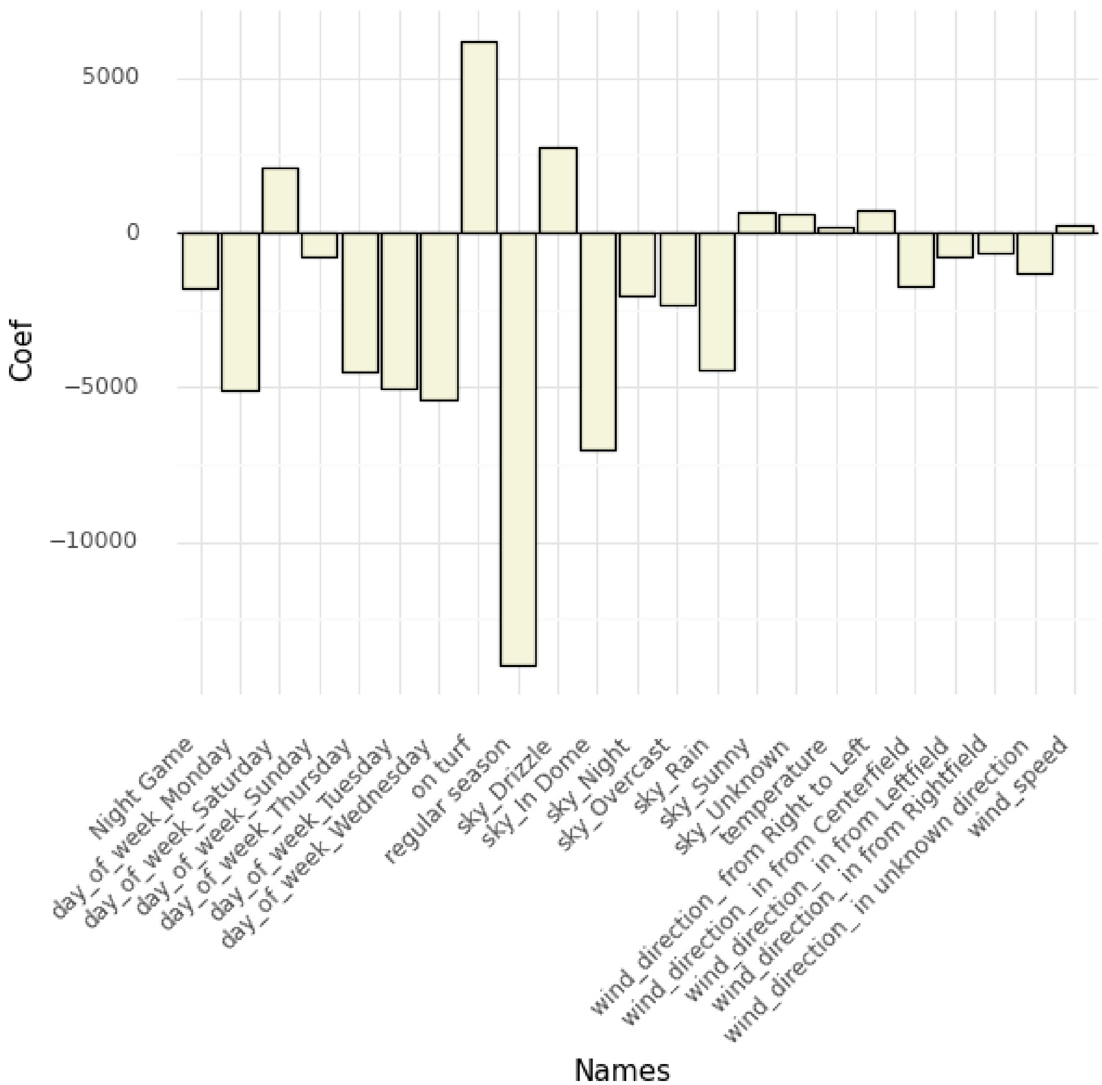
```
X_train[cont] = z.transform(X_train[cont])
X_test[cont] = z.transform(X_test[cont])

lr2 = LinearRegression()
lr2.fit(X_train,y_train)

coef = pd.DataFrame({"Coef": lr2.coef_, "Names": q6preds})
coef
```

[ ]:

	Coef	Names
0	252.303431	wind_speed
1	152.449277	temperature
2	-1802.718132	Night Game
3	-13955.147975	regular season
4	6192.711479	on turf
5	-5116.636271	day_of_week_Monday
6	2090.068654	day_of_week_Saturday
7	-810.303959	day_of_week_Sunday
8	-4482.313767	day_of_week_Thursday
9	-5045.168892	day_of_week_Tuesday
10	-5373.172784	day_of_week_Wednesday
11	2741.553154	sky_Drizzle
12	-7045.339449	sky_In Dome
13	-2063.096631	sky_Night
14	-2335.307746	sky_Overcast
15	-4437.905233	sky_Rain
16	667.993666	sky_Sunny
17	614.568548	sky_Unknown
18	711.573036	wind_direction_ from Right to Left
19	-1747.726921	wind_direction_ in from Centerfield
20	-787.183504	wind_direction_ in from Leftfield
21	-653.702680	wind_direction_ in from Rightfield
22	-1311.215251	wind_direction_ in unknown direction
23	12303.531398	venue_AT&T Park
24	8822.448111	venue_Angel Stadium of Anaheim
25	13878.236051	venue_Busch Stadium III
26	3162.338430	venue_Chase Field
27	6519.984085	venue_Citi Field
28	-4783.265469	venue_Citizens Bank Park
29	2724.157336	venue_Comerica Park
30	4332.399687	venue_Coors Field
31	17167.245323	venue_Dodger Stadium
32	8003.158167	venue_Fenway Park
33	5981.454025	venue_Globe Life Park in Arlington
34	-3676.527153	venue_Great American Ball Park
35	2930.027204	venue_Kauffman Stadium
36	1319.585059	venue_Marlins Park

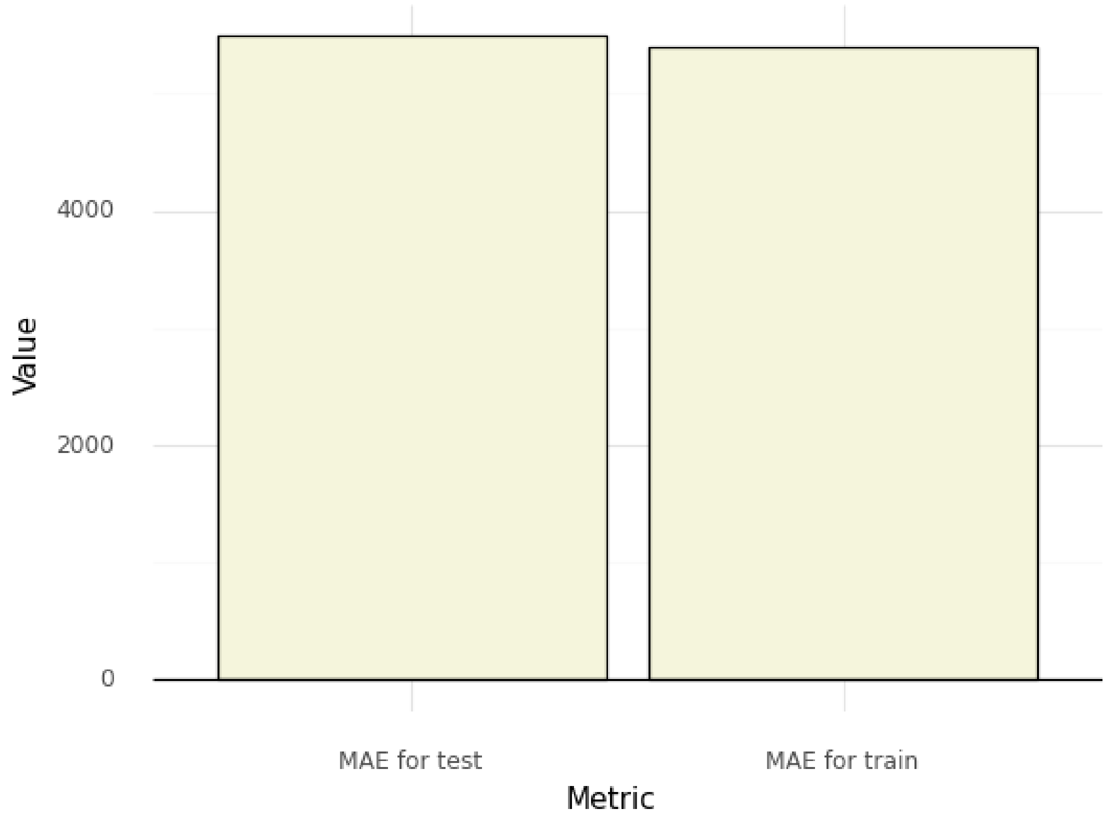
```
37   4031.156514                        venue_Miller Park
38   8777.869930                   venue_Minute Maid Park
39   3153.029819                     venue_Nationals Park
40 -10219.967699  venue_Oakland-Alameda County Coliseum
41  -1567.274919      venue_Oriole Park at Camden Yards
42    385.908908                           venue_PNC Park
43    501.439667                         venue_Petco Park
```

[ ]:
```
(ggplot(coef.head(23), aes(x = 'Names', y = 'Coef'))
+ geom_bar(stat = 'identity', color = 'black', fill = 'beige')
+ theme_minimal()
+ theme(axis_text_x=element_text(angle=45, ha='right'))
+ geom_hline(yintercept = 0))
```



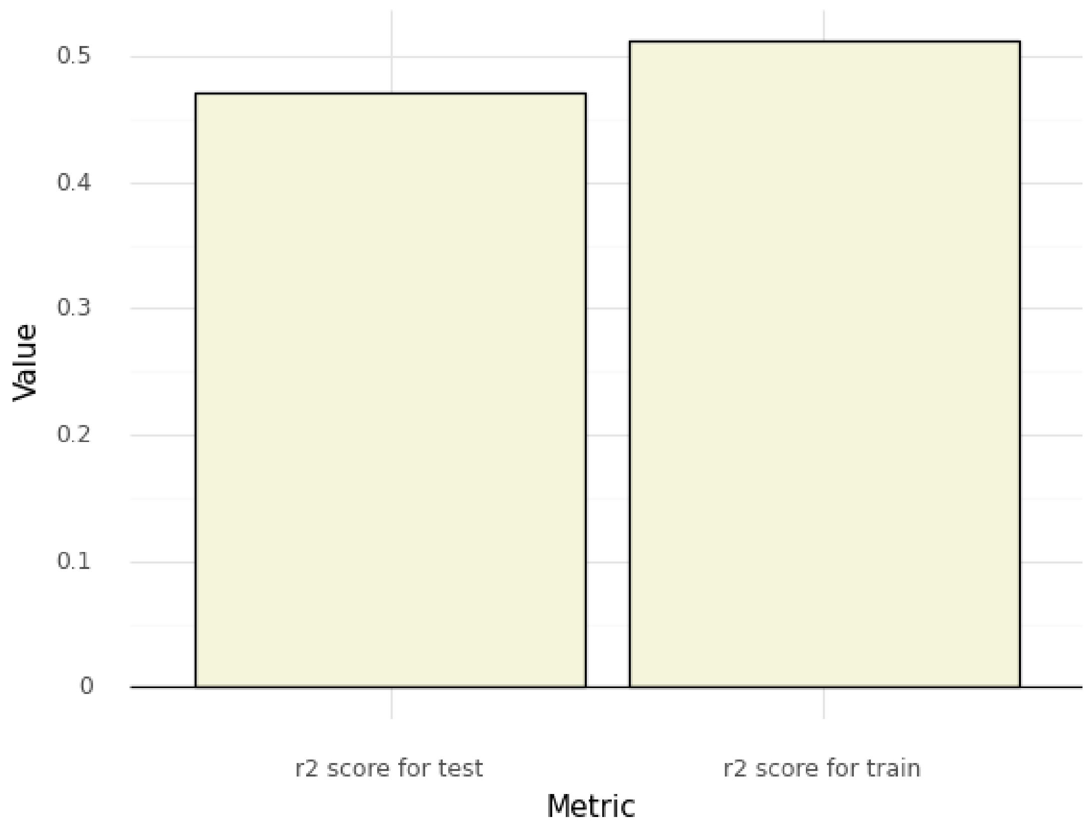
[ ]: <ggplot: (8757944022004)>

[ ]:
```
y_pred = lr2.predict(X_test)
lrMaeTrain2 = mean_absolute_error(y_train, lr2.predict(X_train))
lrMaeTest2 = mean_absolute_error(y_test, y_pred)

lrR2Train2 = r2_score(y_train, lr2.predict(X_train))
lrR2Test2 = r2_score(y_test, lr2.predict(X_test))

values2 = {
    'Metric': ['MAE for train', 'MAE for test', 'r2 score for train', 'r2 score␣
 ↪for test'],
    'Value': [lrMaeTrain2, lrMaeTest2, lrR2Train2, lrR2Test2]
}

values3 = {
   'Metric' : ['r2 score for train', 'r2 score for test'],
   'Value': [lrR2Train2, lrR2Test2]

}
dfq6error = pd.DataFrame(values2)
dfq6errorr2 = pd.DataFrame(values3)
(ggplot(dfq6error.head(2), aes(x = 'Metric', y = 'Value'))
 + geom_bar(stat = 'identity', color = 'black', fill = 'beige')
 + theme_minimal()
 + geom_hline(yintercept = 0))
```




```
[ ]: <ggplot: (8757941720698)>
```

```
[ ]: (ggplot(dfq6errorr2, aes(x = 'Metric', y = 'Value'))
      + geom_bar(stat = 'identity', color = 'black', fill = 'beige')
      + theme_minimal()
      + geom_hline(yintercept = 0))
```



[ ]: <ggplot: (8757941724401)>

0.11 Much to my surprise, and disappointment, envorinmental factors do not explain much of the variance found in attendance numbers for games in this dataset. However, when analyzing the model it was still interesting to see the rather massive effect playoff games have as opposed to regular season games. However, it did not shock me to see the coefficient of games played on saturday having a positive impact, as it makes sense that more people will be free on the weekend.

0.12 Question 7. How heavily is away_runs_scored related to the home_team_win variable?

[ ]:
```
q7feats = ['away_team_errors', 'away_team_hits',
       'away_team_runs',
       'home_team_errors', 'home_team_hits',
       'temperature', 'wind_speed',
        'game_hours_dec',
        'Night Game','on turf', 'home_team_win']
q7preds = ['away_team_errors', 'away_team_hits',
```

```
        'away_team_runs',
        'home_team_errors', 'home_team_hits',
        'temperature', 'wind_speed',
         'game_hours_dec',
         'Night Game','on turf']
q7cont = ['away_team_errors', 'away_team_hits',
        'away_team_runs',
        'home_team_errors', 'home_team_hits',
        'temperature', 'wind_speed',
         'game_hours_dec']
q7df = data[q7feats]

q7df.head()
```

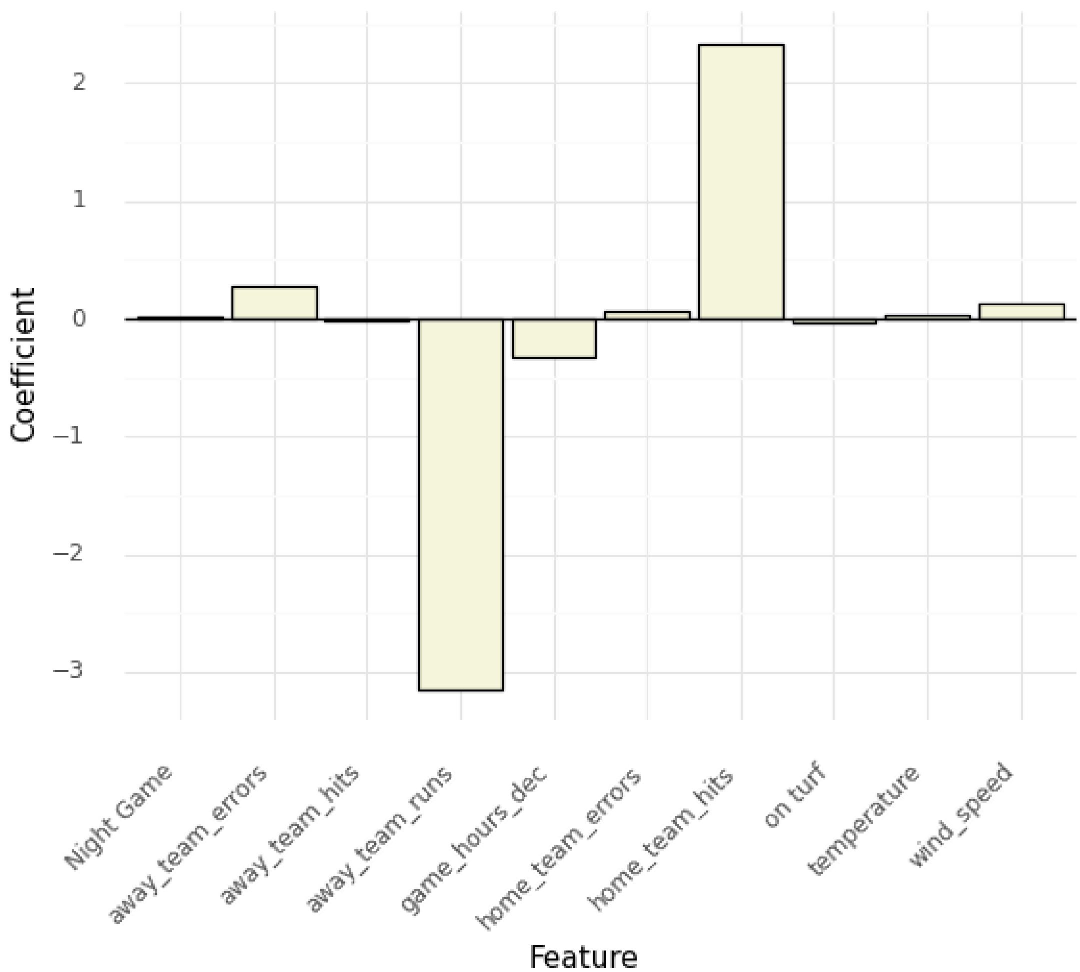
[ ]:

```
   away_team_errors  away_team_hits  away_team_runs  home_team_errors  \
0                 1               7               3                 0
1                 0               5               2                 0
2                 0               5               2                 0
3                 0               8               3                 1
4                 1               8               4                 0

   home_team_hits  temperature  wind_speed  game_hours_dec  Night Game  \
0               9         74.0        14.0        3.216667           1
1               8         55.0        24.0        2.383333           1
2               9         48.0         7.0        3.183333           1
3               8         65.0        10.0        2.883333           1
4               8         77.0         0.0        2.650000           0

   on turf  home_team_win
0        0              1
1        0              1
2        0              1
3        0              0
4        0              0
```

[ ]:

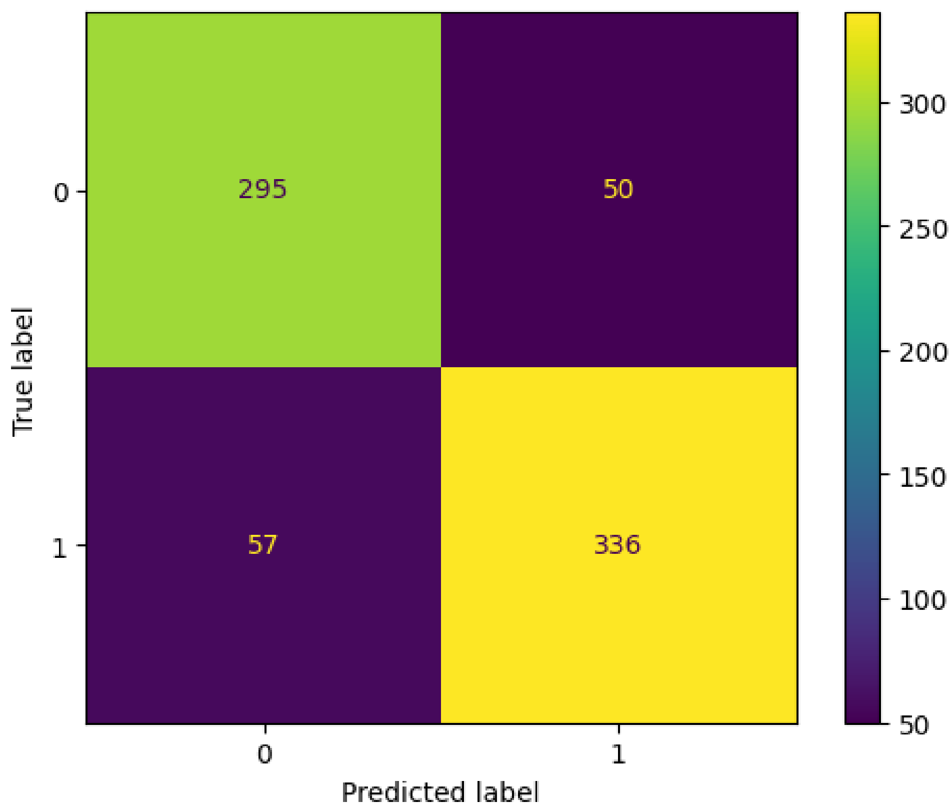
```
X = q7df[q7preds]
y = q7df['home_team_win']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
 ↪random_state=392)

z = StandardScaler()
X_train[q7cont] = z.fit_transform(X_train[q7cont])
X_test[q7cont] = z.transform(X_test[q7cont])

logit2 = LogisticRegression()
logit2.fit(X_train, y_train)
```

```
coefficients = logit2.coef_
coefficients_df = pd.DataFrame({'Feature': q7preds, 'Coefficient': coefficients.
↪flatten()})

coefficients_df['Odds Ratio'] = np.exp(coefficients_df['Coefficient'])



(ggplot(coefficients_df, aes(x = 'Feature', y = 'Coefficient'))
+ geom_bar(stat = 'identity', position = 'identity', color = 'black', fill =␣
↪'beige')
+ theme_minimal()
+ theme(axis_text_x=element_text(angle=45, ha='right'))
+ geom_hline(yintercept = 0))
```



[ ]: <ggplot: (8757941640406)>

[ ]:
```
predictedVals = logit2.predict(X_test) #predict
predictedProbs = logit2.predict_proba(X_test)

pd.DataFrame({"Accuracy: ": accuracy_score(y_test, predictedVals),
"F1 Score: ": f1_score(y_test, predictedVals),
"Recall: ": recall_score(y_test, predictedVals),
"Precision: " : precision_score(y_test, predictedVals)}, index = [0])

matrix = confusion_matrix(y_test, predictedVals)
disp = ConfusionMatrixDisplay(matrix)
disp.plot()
plt.show()
```



[ ]:
```
pd.DataFrame({"Accuracy: ": accuracy_score(y_test, predictedVals),
"F1 Score: ": f1_score(y_test, predictedVals),
"Recall: ": recall_score(y_test, predictedVals),
"Precision: " : precision_score(y_test, predictedVals)}, index = [0])
```

[ ]:
```
   Accuracy:  F1 Score:  Recall:  Precision:
0   0.855014   0.862644  0.854962   0.870466
```

0.13 When comparing this confusion matrix to the one we created earlier in this project, NOT including away_team_runs, This new one has a significantly higher performance. Those with knowledge of baseball or sports in general can understand why this is, as typically the more the road team scores, the less of a chance the home team has of winning.

[ ]:
```
# 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 "/content/drive/MyDrive/FinalProject392JustinLewinski.ipynb" ./

# Again, replace "Class6-Completed.ipynb" to whatever your file is called (see␣
 ↪top of notebook)
!jupyter nbconvert --to PDF "FinalProject392JustinLewinski.ipynb"
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
pandoc is already the newest version (2.5-3build2).
texlive is already the newest version (2019.20200218-1).
texlive-latex-extra is already the newest version (2019.202000218-1).
texlive-xetex is already the newest version (2019.20200218-1).
0 upgraded, 0 newly installed, 0 to remove and 24 not upgraded.
Hit:1 https://cloud.r-project.org/bin/linux/ubuntu focal-cran40/ InRelease
Hit:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2004/x86_64
InRelease
Hit:3 http://ppa.launchpad.net/c2d4u.team/c2d4u4.0+/ubuntu focal InRelease
Hit:4 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:5 http://archive.ubuntu.com/ubuntu focal InRelease
```

```
Hit:6 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:7 http://ppa.launchpad.net/cran/libgit2/ubuntu focal InRelease
Hit:8 http://archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:9 http://ppa.launchpad.net/deadsnakes/ppa/ubuntu focal InRelease
Hit:10 http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu focal InRelease
Hit:11 http://ppa.launchpad.net/ubuntugis/ppa/ubuntu focal InRelease
Reading package lists… Done
Reading package lists… Done
Building dependency tree
Reading state information… Done
texlive-fonts-recommended is already the newest version (2019.20200218-1).
texlive-plain-generic is already the newest version (2019.202000218-1).
texlive-xetex is already the newest version (2019.20200218-1).
0 upgraded, 0 newly installed, 0 to remove and 24 not upgraded.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: pypandoc in /usr/local/lib/python3.10/dist-
packages (1.11)
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
[NbConvertApp] Converting notebook FinalProject392JustinLewinski.ipynb to PDF
[NbConvertApp] Support files will be in FinalProject392JustinLewinski_files/
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Making directory ./FinalProject392JustinLewinski_files
[NbConvertApp] Writing 144010 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
```