Basketball_project

July 2, 2021

1 Prediction on Scores Scored by UC Basketball Players

1.1 Python(NumPy, Pandas, Seaborn, matplotlib, sklearn, OneHotEncoder, GridSearchCV ,KFold), Machine Learning Models(OLS Regression, Decision Tree Regressor, Random Forest Regressor, Lasso Regression, Ridge Regression), Jupyter Notebook

[1]:
```
import re
import nltk
import time
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
import statsmodels.formula.api as smf
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import KFold
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

[2]:
```
import warnings
warnings.simplefilter('ignore')
warnings.filterwarnings('ignore')
```

2 Data Cleaning

[3]:
```
#function that reads position dataframe
def position_df(html):
    table = pd.read_html(html)
    table = table[2]
    return table
```

[4]:
```
ucb21_pos = position_df('https://calbears.com/sports/mens-basketball/roster/
↪2020-21')
ucb20_pos = position_df('https://calbears.com/sports/mens-basketball/roster/
↪2019-20')
ucb19_pos = position_df('https://calbears.com/sports/mens-basketball/roster/
↪2018-19')
ucb18_pos = position_df('https://calbears.com/sports/mens-basketball/roster/
↪2017-18')
ucb17_pos = position_df('https://calbears.com/sports/mens-basketball/roster/
↪2016-17')
ucb16_pos = position_df('https://calbears.com/sports/mens-basketball/roster/
↪2015-16')
ucb14_pos = position_df('https://calbears.com/sports/mens-basketball/roster/
↪2013-14')

ucsd21_pos = position_df('https://ucsdtritons.com/sports/mens-basketball/roster/
↪2020-21')
ucsd20_pos = position_df('https://ucsdtritons.com/sports/mens-basketball/roster/
↪2019-20')
ucsd19_pos = position_df('https://ucsdtritons.com/sports/mens-basketball/roster/
↪2018-19')
ucsd18_pos = position_df('https://ucsdtritons.com/sports/mens-basketball/roster/
↪2017-18')
ucsd17_pos = position_df('https://ucsdtritons.com/sports/mens-basketball/roster/
↪2016-17')

uci21_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
↪roster/2020-21')
uci20_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
↪roster/2019-20')
uci19_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
↪roster/2018-19')
uci18_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
↪roster/2017-18')
uci17_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
↪roster/2016-17')
uci16_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
↪roster/2015-16')
```

```
uci15_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
 ↪roster/2014-15')
uci14_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
 ↪roster/2013-14')
uci13_pos = position_df('https://ucirvinesports.com/sports/mens-basketball/
 ↪roster/2012-13')

ucla21_pos = position_df('https://uclabruins.com/sports/mens-basketball/roster/
 ↪2020-21')
ucla20_pos = position_df('https://uclabruins.com/sports/mens-basketball/roster/
 ↪2019-20')
ucla19_pos = position_df('https://uclabruins.com/sports/mens-basketball/roster/
 ↪2018-19')
ucla18_pos = position_df('https://uclabruins.com/sports/mens-basketball/roster/
 ↪2017-18')
ucla17_pos = position_df('https://uclabruins.com/sports/mens-basketball/roster/
 ↪2016-17')
ucla16_pos = position_df('https://uclabruins.com/sports/mens-basketball/roster/
 ↪2015-16')

ucd21_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2020-21')
ucd20_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2019-20')
ucd19_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2018-19')
ucd18_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2017-18')
ucd17_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2016-17')
ucd16_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2015-16')
ucd15_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2014-15')
ucd14_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2013-14')
ucd13_pos = position_df('https://ucdavisaggies.com/sports/mens-basketball/
 ↪roster/2012-13')
```

[5]:
```
#function that reads statistics dataframe
def stats_df(html):
    table = pd.read_html(html, match = 'Player Averages')
    table = table[0]
    table = table.droplevel(0, axis=1)
    table = table.drop(table.index[len(table)-2]).drop(table.
 ↪index[len(table)-1])
```

```
    return table
```

[6]:
```
ucb21_st = stats_df('https://calbears.com/sports/mens-basketball/stats/
↪2020-21#individual')
ucb20_st = stats_df('https://calbears.com/sports/mens-basketball/stats/
↪2019-20#individual')
ucb19_st = stats_df('https://calbears.com/sports/mens-basketball/stats/
↪2018-19#individual')
ucb18_st = stats_df('https://calbears.com/sports/mens-basketball/stats/
↪2017-18#individual')
ucb17_st = stats_df('https://calbears.com/sports/mens-basketball/stats/
↪2016-17#individual')
ucb16_st = stats_df('https://calbears.com/sports/mens-basketball/stats/
↪2015-16#individual')
ucb14_st = stats_df('https://calbears.com/sports/mens-basketball/stats/
↪2013-14#individual')

ucsd21_st = stats_df('https://ucsdtritons.com/sports/mens-basketball/stats/
↪2020-21#individual')
ucsd20_st = stats_df('https://ucsdtritons.com/sports/mens-basketball/stats/
↪2019-20#individual')
ucsd19_st = stats_df('https://ucsdtritons.com/sports/mens-basketball/stats/
↪2018-19#individual')
ucsd18_st = stats_df('https://ucsdtritons.com/sports/mens-basketball/stats/
↪2017-18#individual')
ucsd17_st = stats_df('https://ucsdtritons.com/sports/mens-basketball/stats/
↪2016-17#individual')

uci21_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2020-21#individual')
uci20_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2019-20#individual')
uci19_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2018-19#individual')
uci18_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2017-18#individual')
uci17_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2016-17#individual')
uci16_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2015-16#individual')
uci15_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2014-15#individual')
uci14_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2013-14#individual')
```

```
uci13_st = stats_df('https://ucirvinesports.com/sports/mens-basketball/stats/
↪2012-13#individual')

ucla21_st = stats_df('https://uclabruins.com/sports/mens-basketball/stats/
↪2020-21#individual')
ucla20_st = stats_df('https://uclabruins.com/sports/mens-basketball/stats/
↪2019-20#individual')
ucla19_st = stats_df('https://uclabruins.com/sports/mens-basketball/stats/
↪2018-19#individual')
ucla18_st = stats_df('https://uclabruins.com/sports/mens-basketball/stats/
↪2017-18#individual')
ucla17_st = stats_df('https://uclabruins.com/sports/mens-basketball/stats/
↪2016-17#individual')
ucla16_st = stats_df('https://uclabruins.com/sports/mens-basketball/stats/
↪2015-16#individual')

ucd21_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2020-21#individual')
ucd20_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2019-20#individual')
ucd19_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2018-19#individual')
ucd18_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2017-18#individual')
ucd17_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2016-17#individual')
ucd16_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2015-16#individual')
ucd15_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2014-15#individual')
ucd14_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2013-14#individual')
ucd13_st = stats_df('https://ucdavisaggies.com/sports/mens-basketball/stats/
↪2012-13#individual')
```

[7]:
```
#function that change column name
def change_colname(df, old, new):
    df = df.rename(columns={old: new})
    return df
```

[8]:
```
#function that change the column names so that all the df have same column names
ucb21_st = change_colname(ucb21_st, "#", "No.")
ucb20_st = change_colname(ucb20_st, "#", "No.")
ucb19_st = change_colname(ucb19_st, "#", "No.")
ucb18_st = change_colname(ucb18_st, "#", "No.")
ucb17_st = change_colname(ucb17_st, "#", "No.")
```

```
ucb16_st = change_colname(ucb16_st, "#", "No.")
ucb14_st = change_colname(ucb14_st, "#", "No.")

ucsd21_st = change_colname(ucsd21_st, "#", "No.")
ucsd20_st = change_colname(ucsd20_st, "#", "No.")
ucsd19_st = change_colname(ucsd19_st, "#", "No.")
ucsd18_st = change_colname(ucsd18_st, "#", "No.")
ucsd17_st = change_colname(ucsd17_st, "#", "No.")

uci21_st = change_colname(uci21_st, "#", "No.")
uci20_st = change_colname(uci20_st, "#", "No.")
uci19_st = change_colname(uci19_st, "#", "No.")
uci18_st = change_colname(uci18_st, "#", "No.")
uci17_st = change_colname(uci17_st, "#", "No.")
uci16_st = change_colname(uci16_st, "#", "No.")
uci15_st = change_colname(uci15_st, "#", "No.")
uci14_st = change_colname(uci14_st, "#", "No.")
uci13_st = change_colname(uci13_st, "#", "No.")

ucla21_st = change_colname(ucla21_st, "#", "No.")
ucla20_st = change_colname(ucla20_st, "#", "No.")
ucla19_st = change_colname(ucla19_st, "#", "No.")
ucla18_st = change_colname(ucla18_st, "#", "No.")
ucla17_st = change_colname(ucla17_st, "#", "No.")
ucla16_st = change_colname(ucla16_st, "#", "No.")

ucd21_st = change_colname(ucd21_st, "#", "No.")
ucd20_st = change_colname(ucd20_st, "#", "No.")
ucd19_st = change_colname(ucd19_st, "#", "No.")
ucd18_st = change_colname(ucd18_st, "#", "No.")
ucd17_st = change_colname(ucd17_st, "#", "No.")
ucd16_st = change_colname(ucd16_st, "#", "No.")
ucd15_st = change_colname(ucd15_st, "#", "No.")
ucd14_st = change_colname(ucd14_st, "#", "No.")
ucd13_st = change_colname(ucd13_st, "#", "No.")
```

[9]:
```
ucsd19_pos = change_colname(ucsd19_pos, "#", "No.")
ucsd18_pos = change_colname(ucsd18_pos, "#", "No.")
ucsd17_pos = change_colname(ucsd17_pos, "#", "No.")

ucb21_pos = change_colname(ucb21_pos, "#", "No.")
ucb20_pos = change_colname(ucb20_pos, "#", "No.")
ucb19_pos = change_colname(ucb19_pos, "#", "No.")
ucb18_pos = change_colname(ucb18_pos, "#", "No.")
ucb17_pos = change_colname(ucb17_pos, "#", "No.")
ucb16_pos = change_colname(ucb16_pos, "#", "No.")
ucb14_pos = change_colname(ucb14_pos, "#", "No.")
```

```
uci21_pos = change_colname(uci21_pos, "#", "No.")
uci20_pos = change_colname(uci20_pos, "#", "No.")
uci19_pos = change_colname(uci19_pos, "#", "No.")
uci18_pos = change_colname(uci18_pos, "#", "No.")
uci17_pos = change_colname(uci17_pos, "#", "No.")
uci16_pos = change_colname(uci16_pos, "#", "No.")
uci15_pos = change_colname(uci15_pos, "#", "No.")
uci14_pos = change_colname(uci14_pos, "#", "No.")
uci13_pos = change_colname(uci13_pos, "#", "No.")

ucla21_pos = change_colname(ucla21_pos, "#", "No.")
ucla20_pos = change_colname(ucla20_pos, "#", "No.")
ucla19_pos = change_colname(ucla19_pos, "#", "No.")
ucla18_pos = change_colname(ucla18_pos, "#", "No.")
ucla17_pos = change_colname(ucla17_pos, "#", "No.")
ucla16_pos = change_colname(ucla16_pos, "#", "No.")

ucd21_pos = change_colname(ucd21_pos, "#", "No.")
ucd20_pos = change_colname(ucd20_pos, "#", "No.")
ucd19_pos = change_colname(ucd19_pos, "#", "No.")
ucd18_pos = change_colname(ucd18_pos, "#", "No.")
ucd17_pos = change_colname(ucd17_pos, "#", "No.")
ucd16_pos = change_colname(ucd16_pos, "#", "No.")
ucd15_pos = change_colname(ucd15_pos, "#", "No.")
ucd14_pos = change_colname(ucd14_pos, "#", "No.")
ucd13_pos = change_colname(ucd13_pos, "#", "No.")

ucsd21_pos = change_colname(ucsd21_pos, "Name", "Full Name")
ucsd20_pos = change_colname(ucsd20_pos, "Name", "Full Name")
ucsd19_pos = change_colname(ucsd19_pos, "Name", "Full Name")
ucsd18_pos = change_colname(ucsd18_pos, "Name", "Full Name")
ucsd17_pos = change_colname(ucsd17_pos, "Name", "Full Name")

ucsd21_pos = change_colname(ucsd21_pos, "Pos", "Pos.")
ucsd20_pos = change_colname(ucsd20_pos, "Pos", "Pos.")
ucsd19_pos = change_colname(ucsd19_pos, "Pos", "Pos.")
ucsd18_pos = change_colname(ucsd18_pos, "Pos", "Pos.")
ucsd17_pos = change_colname(ucsd17_pos, "Pos", "Pos.")

uci19_pos = change_colname(uci19_pos, "Pos", "Pos.")
uci18_pos = change_colname(uci18_pos, "Pos", "Pos.")
uci17_pos = change_colname(uci17_pos, "Pos", "Pos.")
uci16_pos = change_colname(uci16_pos, "Pos", "Pos.")
uci15_pos = change_colname(uci15_pos, "Pos", "Pos.")
uci14_pos = change_colname(uci14_pos, "Pos", "Pos.")
uci13_pos = change_colname(uci13_pos, "Pos", "Pos.")
```

```
ucb17_pos = change_colname(ucb17_pos, "Pos", "Pos.")
ucb16_pos = change_colname(ucb16_pos, "Pos", "Pos.")
ucb14_pos = change_colname(ucb14_pos, "Pos", "Pos.")

ucd17_pos = change_colname(ucd17_pos, "Pos", "Pos.")
ucd16_pos = change_colname(ucd16_pos, "Pos", "Pos.")
ucd15_pos = change_colname(ucd15_pos, "Pos", "Pos.")
ucd14_pos = change_colname(ucd14_pos, "Pos", "Pos.")
ucd13_pos = change_colname(ucd13_pos, "Pos", "Pos.")
```

[10]:
```
#function that merge df
#only players who are in both df will be return
def merge_df(df1, df2):
    merge = df1.merge(df2, left_on='No.', right_on='No.')
    return merge
```

[11]:
```
ucb21_merge = merge_df(ucb21_st, ucb21_pos)
ucb20_merge = merge_df(ucb20_st, ucb20_pos)
ucb19_merge = merge_df(ucb19_st, ucb19_pos)
ucb18_merge = merge_df(ucb18_st, ucb18_pos)
ucb17_merge = merge_df(ucb17_st, ucb17_pos)
ucb16_merge = merge_df(ucb16_st, ucb16_pos)
ucb14_merge = merge_df(ucb14_st, ucb14_pos)

ucsd21_merge = merge_df(ucsd21_st, ucsd21_pos)
ucsd20_merge = merge_df(ucsd20_st, ucsd20_pos)
ucsd19_merge = merge_df(ucsd19_st, ucsd19_pos)
ucsd18_merge = merge_df(ucsd18_st, ucsd18_pos)
ucsd17_merge = merge_df(ucsd17_st, ucsd17_pos)

uci21_merge = merge_df(uci21_st, uci21_pos)
uci20_merge = merge_df(uci20_st, uci20_pos)
uci19_merge = merge_df(uci19_st, uci19_pos)
uci18_merge = merge_df(uci18_st, uci18_pos)
uci17_merge = merge_df(uci17_st, uci17_pos)
uci16_merge = merge_df(uci16_st, uci16_pos)
uci15_merge = merge_df(uci15_st, uci15_pos)
uci14_merge = merge_df(uci14_st, uci14_pos)
uci13_merge = merge_df(uci13_st, uci13_pos)

ucla21_merge = merge_df(ucla21_st, ucla21_pos)
ucla20_merge = merge_df(ucla20_st, ucla20_pos)
ucla19_merge = merge_df(ucla19_st, ucla19_pos)
ucla18_merge = merge_df(ucla18_st, ucla18_pos)
ucla17_merge = merge_df(ucla17_st, ucla17_pos)
ucla16_merge = merge_df(ucla16_st, ucla16_pos)
```

```
ucd21_merge = merge_df(ucd21_st, ucd21_pos)
ucd20_merge = merge_df(ucd20_st, ucd20_pos)
ucd19_merge = merge_df(ucd19_st, ucd19_pos)
ucd18_merge = merge_df(ucd18_st, ucd18_pos)
ucd17_merge = merge_df(ucd17_st, ucd17_pos)
ucd16_merge = merge_df(ucd16_st, ucd16_pos)
ucd15_merge = merge_df(ucd15_st, ucd15_pos)
ucd14_merge = merge_df(ucd14_st, ucd14_pos)
ucd13_merge = merge_df(ucd13_st, ucd13_pos)
```

[12]:
```
#Include Year variable to help divide the trainingset and testset
ucb21_merge['Year'] = 2021
ucb20_merge['Year'] = 2020
ucb19_merge['Year'] = 2019
ucb18_merge['Year'] = 2018
ucb17_merge['Year'] = 2017
ucb16_merge['Year'] = 2016
ucb14_merge['Year'] = 2014

ucsd21_merge['Year'] = 2021
ucsd20_merge['Year'] = 2020
ucsd19_merge['Year'] = 2019
ucsd18_merge['Year'] = 2018
ucsd17_merge['Year'] = 2017

uci21_merge['Year'] = 2021
uci20_merge['Year'] = 2020
uci19_merge['Year'] = 2019
uci18_merge['Year'] = 2018
uci17_merge['Year'] = 2017
uci16_merge['Year'] = 2016
uci15_merge['Year'] = 2015
uci14_merge['Year'] = 2014
uci13_merge['Year'] = 2013

ucla21_merge['Year'] = 2021
ucla20_merge['Year'] = 2020
ucla19_merge['Year'] = 2019
ucla18_merge['Year'] = 2018
ucla17_merge['Year'] = 2017
ucla16_merge['Year'] = 2016

ucd21_merge['Year'] = 2021
ucd20_merge['Year'] = 2020
ucd19_merge['Year'] = 2019
ucd18_merge['Year'] = 2018
```

```
ucd17_merge['Year'] = 2017
ucd16_merge['Year'] = 2016
ucd15_merge['Year'] = 2015
ucd14_merge['Year'] = 2014
ucd13_merge['Year'] = 2013
```

[13]:
```
#Change column orders and drop variables that are not needed
ucb21 = ucb21_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucb20 = ucb20_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucb19 = ucb19_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucb18 = ucb18_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucb17 = ucb17_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucb16 = ucb16_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucb14 = ucb14_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]

ucsd21 = ucsd21_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucsd20 = ucsd20_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucsd19 = ucsd19_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucsd18 = ucsd18_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucsd17 = ucsd17_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]

uci21 = uci21_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
```

```
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
uci20 = uci20_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
uci19 = uci19_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
uci18 = uci18_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
uci17 = uci17_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
uci16 = uci16_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
uci15 = uci15_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
uci14 = uci14_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
uci13 = uci13_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]


ucla21 = ucla21_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucla20 = ucla20_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucla19 = ucla19_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucla18 = ucla18_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucla17 = ucla17_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucla16 = ucla16_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                    'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
```

```
ucd21 = ucd21_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucd20 = ucd20_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucd19 = ucd19_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucd18 = ucd18_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucd17 = ucd17_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucd16 = ucd16_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucd15 = ucd15_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucd14 = ucd14_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
ucd13 = ucd13_merge[['Pos.', 'Ht.', 'GP', 'MIN', 'FG%', '3PT%',
                     'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'PTS',␣
 ↪'Year']]
```

[14]:
```
#Combine all the data to one big df
basketball = pd.concat([ucb21, ucb20, ucb19, ucb18, ucb17, ucb16, ucb14,
                        ucsd21, ucsd20, ucsd19, ucsd18, ucsd17,
                        uci21, uci20, uci19, uci18, uci17, uci16, uci15, uci14,␣
 ↪uci13,
                        ucla21, ucla20, ucla19, ucla18, ucla17, ucla16,
                        ucd21, ucd20, ucd19, ucd18, ucd17, ucd16, ucd15, ucd14,␣
 ↪ucd13])
basketball.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 518 entries, 0 to 12
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Pos.    518 non-null    object
 1   Ht.     518 non-null    object
 2   GP      518 non-null    float64
```

```
 3   MIN     518 non-null    float64
 4   FG%     518 non-null    float64
 5   3PT%    518 non-null    float64
 6   FT%     518 non-null    float64
 7   OREB    518 non-null    float64
 8   DREB    518 non-null    float64
 9   REB     518 non-null    float64
 10  AST     518 non-null    float64
 11  STL     518 non-null    float64
 12  BLK     518 non-null    float64
 13  PTS     518 non-null    float64
 14  Year    518 non-null    int64
dtypes: float64(12), int64(1), object(2)
memory usage: 64.8+ KB
```

[15]:
```
basketball.replace("-", float("NaN"), inplace=True)
basketball.dropna(subset = ["Ht."], inplace=True)
basketball.dropna(inplace = True)
basketball.shape
```

[15]:
```
(515, 15)
```

[16]:
```
#Converting Ht. to inches
#New column for Height
new = basketball["Ht."].str.split("-", n = 1, expand = True)

basketball["ft"]= new[0]
basketball["inch"]= new[1]

basketball["ft"] = basketball["ft"].astype(float, errors = 'raise')
basketball["inch"] = basketball["inch"].astype(float, errors = 'raise')
basketball["Height"] = basketball["ft"] * 12 + basketball["inch"]

basketball.drop(columns =["Ht.", "ft", "inch"], inplace = True)
```

[17]:
```
#Replace values in Pos. column to have constant values
basketball['Pos.'] = basketball['Pos.'].replace({'Forward':'F', 'Guard':'G',␣
 ↪'Center':'C',
                                                 'Forward/Center':'F/C',␣
 ↪'Forward/Guard':'G/F',
                                                 'C/F':'F/C'})
```

[18]:
```
basketball
```

[18]:

	Pos.	GP	MIN	FG%	3PT%	FT%	OREB	DREB	REB	AST	STL	BLK	\
0	G	22.0	30.2	0.455	0.364	0.821	1.0	3.6	4.6	1.7	0.4	0.2	
1	F	29.0	25.9	0.587	0.000	0.590	1.8	4.7	6.4	0.6	0.6	0.5	

```
2     F  25.0  29.5  0.408  0.366  0.857   0.8   3.8  4.5  1.8  0.5  0.4
3     G  29.0  27.3  0.354  0.327  0.800   0.3   3.0  3.4  1.2  0.3  0.3
4     G  29.0  20.5  0.332  0.317  0.868   0.2   1.0  1.2  1.6  0.5  0.0
..  …   …   …    …    …    …   …   …   …  …  …  …  …
8     F  20.0   7.9  0.531  0.000  0.400   0.5   0.9  1.4  0.4  0.4  0.4
9     F   4.0   2.5  0.333  0.286  0.000   0.0   0.8  0.8  0.0  0.3  0.0
10    C  12.0   6.3  0.364  0.000  0.800   0.5   0.8  1.3  0.1  0.1  0.0
11    G  24.0   8.8  0.421  0.250  0.429   0.2   0.5  0.6  0.5  0.3  0.0
12    F   2.0   1.5  0.000  0.000  0.000   0.5   0.5  1.0  0.5  0.5  0.0

     PTS  Year  Height
0   18.0  2021    76.0
1   10.3  2021    80.0
2    8.9  2021    80.0
3    8.5  2021    77.0
4    7.2  2021    73.0
..   …   …     …
8    2.0  2013    80.0
9    2.0  2013    77.0
10   1.0  2013    82.0
11   0.9  2013    74.0
12   0.0  2013    80.0

[515 rows x 15 columns]
```

[19]:
```
#Split df into trainingset and testset
#trainingset: 72.6%
#testset: 27.4%
#the most recently observed data are in testset
basketballtrain = basketball[basketball['Year'] <= 2019]
basketballtest = basketball[basketball['Year'] > 2019]
basketballtrain.drop(['Year'], axis=1, inplace=True)
basketballtest.drop(['Year'], axis=1, inplace=True)

len(basketballtrain), len(basketballtest)
```

[19]:
```
(374, 141)
```

[20]:
```
basketballtest
```

[20]:
```
   Pos.    GP   MIN    FG%   3PT%    FT%  OREB  DREB  REB  AST  STL  BLK  \
0     G  22.0  30.2  0.455  0.364  0.821   1.0   3.6  4.6  1.7  0.4  0.2
1     F  29.0  25.9  0.587  0.000  0.590   1.8   4.7  6.4  0.6  0.6  0.5
2     F  25.0  29.5  0.408  0.366  0.857   0.8   3.8  4.5  1.8  0.5  0.4
3     G  29.0  27.3  0.354  0.327  0.800   0.3   3.0  3.4  1.2  0.3  0.3
4     G  29.0  20.5  0.332  0.317  0.868   0.2   1.0  1.2  1.6  0.5  0.0
..  …   …   …    …    …    …   …   …   …  …  …  …  …
```

```
7     F  32.0  19.8  0.448  0.286  0.731   1.3   2.8  4.1  0.9  0.6  0.3
8     G  22.0  16.5  0.415  0.390  0.900   0.5   2.0  2.5  0.7  0.5  0.1
9     G   5.0   4.0  0.500  0.571  0.000   0.4   1.0  1.4  0.2  0.0  0.2
10    G  10.0   4.4  0.500  0.250  0.000   0.1   0.6  0.7  0.2  0.1  0.0
11    G  29.0   9.0  0.500  0.167  1.000   0.1   0.6  0.7  0.4  0.1  0.3

     PTS  Height
0   18.0    76.0
1   10.3    80.0
2    8.9    80.0
3    8.5    77.0
4    7.2    73.0
..   …      …
7    5.8    79.0
8    3.6    76.0
9    2.4    75.0
10   1.5    78.0
11   1.0    76.0

[141 rows x 14 columns]
```

[21]:
```
#Below is the code for making dummies variable
basketballtrain_dumm = pd.get_dummies(basketballtrain, columns = ['Pos.'],␣
 ↪drop_first = True)
basketballtrain_dumm

basketballtest_dumm = pd.get_dummies(basketballtest, columns = ['Pos.'],␣
 ↪drop_first = True)
basketballtest_dumm.columns
```

[21]:
```
Index(['GP', 'MIN', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB', 'REB', 'AST', 'STL',
       'BLK', 'PTS', 'Height', 'Pos._F', 'Pos._F/C', 'Pos._G', 'Pos._G/F'],
      dtype='object')
```

[22]:
```
y_train = basketballtrain['PTS']
X_train = basketballtrain.drop(['PTS'], axis=1)

y_test = basketballtest['PTS']
X_test = basketballtest.drop(['PTS'], axis=1)
```

3 1. Linear Regression

[23]:
```
def OSR2(model, X_test, y_test, y_train):
    y_pred = model.predict(X_test)
    SSE = np.sum((y_test - y_pred)**2)
    SST = np.sum((y_test - np.mean(y_train))**2)
```

```
    return (1 - SSE/SST)

def VIF(data, columns):
    values = sm.add_constant(data[columns]).values
    num_col = len(columns) + 1
    vif = [variance_inflation_factor(values, i) for i in range(num_col)]
    return pd.Series(vif[1:], index=columns)
```

[24]:
```
cols = ['Height', 'GP', 'MIN', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB', 'REB',
       'AST', 'STL', 'BLK','Pos._F', 'Pos._F/C', 'Pos._G',
       'Pos._G/F']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

X_ = sm.add_constant(X_)
X_t = sm.add_constant(X_t)

model_1 = sm.OLS(y_, X_).fit()
print(model_1.summary())
print('OSR2:', OSR2(model_1, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    PTS   R-squared:                       0.803
Model:                            OLS   Adj. R-squared:                  0.794
Method:                 Least Squares   F-statistic:                     90.99
Date:                Fri, 02 Jul 2021   Prob (F-statistic):          3.17e-115
Time:                        21:26:30   Log-Likelihood:                -832.85
No. Observations:                 374   AIC:                             1700.
Df Residuals:                     357   BIC:                             1766.
Df Model:                          16
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -6.9593      4.605     -1.511      0.132     -16.015       2.096
Height         0.0620      0.056      1.098      0.273      -0.049       0.173
GP            -0.0440      0.013     -3.387      0.001      -0.070      -0.018
MIN            0.2498      0.027      9.365      0.000       0.197       0.302
FG%            1.9071      0.859      2.219      0.027       0.217       3.597
3PT%           3.3491      0.760      4.409      0.000       1.855       4.843
FT%            1.2298      0.532      2.310      0.021       0.183       2.277
OREB          -2.4382      2.740     -0.890      0.374      -7.828       2.951
```

```
DREB          -2.0695      2.730     -0.758      0.449      -7.438       3.299
REB            2.7849      2.721      1.023      0.307      -2.567       8.137
AST            0.5859      0.167      3.502      0.001       0.257       0.915
STL            0.5010      0.391      1.281      0.201      -0.268       1.270
BLK           -0.2682      0.385     -0.697      0.486      -1.025       0.488
Pos._F        -0.3516      0.504     -0.698      0.486      -1.342       0.639
Pos._F/C       0.3510      0.868      0.404      0.686      -1.356       2.058
Pos._G         0.1473      0.658      0.224      0.823      -1.146       1.440
Pos._G/F       0.6383      0.932      0.685      0.494      -1.194       2.471
==============================================================================
Omnibus:                       24.370   Durbin-Watson:                   1.411
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               73.525
Skew:                           0.175   Prob(JB):                     1.08e-16
Kurtosis:                       5.144   Cond. No.                     3.32e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.32e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
OSR2: 0.7837607629987315
```

[24]:
```
Height         3.561254
GP             1.634931
MIN            5.345046
FG%            1.428094
3PT%           1.625312
FT%            1.521782
OREB         296.765284
DREB        1338.144224
REB         2522.172467
AST            3.044997
STL            2.247845
BLK            2.624089
Pos._F         3.775384
Pos._F/C       1.254944
Pos._G         7.581087
Pos._G/F       1.446398
dtype: float64
```

4 1.1 Variable Selection

[25]:
```
#REB - removed / had the largest VIF
cols = ['Height', 'GP', 'MIN', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL', 'BLK','Pos._F', 'Pos._F/C', 'Pos._G',
       'Pos._G/F']
```

```
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

X_ = sm.add_constant(X_)
X_t = sm.add_constant(X_t)

model_2 = sm.OLS(y_, X_).fit()
print(model_2.summary())
print('OSR2:', OSR2(model_2, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    PTS   R-squared:                       0.802
Model:                            OLS   Adj. R-squared:                  0.794
Method:                 Least Squares   F-statistic:                     96.97
Date:                Fri, 02 Jul 2021   Prob (F-statistic):          5.30e-116
Time:                        21:26:31   Log-Likelihood:                -833.40
No. Observations:                 374   AIC:                             1699.
Df Residuals:                     358   BIC:                             1762.
Df Model:                          15
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -6.8558      4.604     -1.489      0.137     -15.910       2.198
Height         0.0608      0.056      1.076      0.283      -0.050       0.172
GP            -0.0441      0.013     -3.394      0.001      -0.070      -0.019
MIN            0.2489      0.027      9.336      0.000       0.196       0.301
FG%            1.8748      0.859      2.183      0.030       0.186       3.564
3PT%           3.4247      0.756      4.529      0.000       1.938       4.912
FT%            1.2548      0.532      2.360      0.019       0.209       2.301
OREB           0.3482      0.310      1.123      0.262      -0.262       0.958
DREB           0.7196      0.159      4.517      0.000       0.406       1.033
AST            0.5892      0.167      3.522      0.000       0.260       0.918
STL            0.4890      0.391      1.251      0.212      -0.280       1.258
BLK           -0.2603      0.385     -0.677      0.499      -1.017       0.496
Pos._F        -0.3675      0.503     -0.730      0.466      -1.358       0.623
Pos._F/C       0.2709      0.864      0.313      0.754      -1.429       1.971
Pos._G         0.1225      0.657      0.186      0.852      -1.170       1.415
Pos._G/F       0.6353      0.932      0.682      0.496      -1.197       2.468
==============================================================================
Omnibus:                       23.510   Durbin-Watson:                   1.426
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               70.305
```

```
Skew:                          0.157   Prob(JB):                     5.41e-16
Kurtosis:                      5.101   Cond. No.                     3.24e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
OSR2: 0.7833205039554457
```

[25]:
```
Height      3.559656
GP          1.634840
MIN         5.339167
FG%         1.426169
3PT%        1.609915
FT%         1.518570
OREB        3.800725
DREB        4.555485
AST         3.043877
STL         2.245832
BLK         2.623046
Pos._F      3.771816
Pos._F/C    1.244729
Pos._G      7.570770
Pos._G/F    1.446384
dtype: float64
```

[26]:
```
#Pos._G - removed / had the next largest VIF
cols = ['Height', 'GP', 'MIN', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL', 'BLK','Pos._F', 'Pos._F/C',
       'Pos._G/F']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

X_ = sm.add_constant(X_)
X_t = sm.add_constant(X_t)

model_3 = sm.OLS(y_, X_).fit()
print(model_3.summary())
print('OSR2:', OSR2(model_3, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                            OLS Regression Results
==============================================================================
```

Dep. Variable:	PTS	R-squared:	0.802
Model:	OLS	Adj. R-squared:	0.795
Method:	Least Squares	F-statistic:	104.2
Date:	Fri, 02 Jul 2021	Prob (F-statistic):	5.19e-117
Time:	21:26:31	Log-Likelihood:	-833.42
No. Observations:	374	AIC:	1697.
Df Residuals:	359	BIC:	1756.
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-6.3038	3.520	-1.791	0.074	-13.226	0.618
Height	0.0548	0.046	1.179	0.239	-0.037	0.146
GP	-0.0440	0.013	-3.396	0.001	-0.070	-0.019
MIN	0.2494	0.027	9.404	0.000	0.197	0.302
FG%	1.8652	0.856	2.178	0.030	0.181	3.549
3PT%	3.4550	0.737	4.685	0.000	2.005	4.905
FT%	1.2487	0.530	2.356	0.019	0.206	2.291
OREB	0.3453	0.309	1.116	0.265	-0.263	0.954
DREB	0.7164	0.158	4.530	0.000	0.405	1.027
AST	0.5873	0.167	3.522	0.000	0.259	0.915
STL	0.5011	0.385	1.301	0.194	-0.256	1.258
BLK	-0.2641	0.384	-0.688	0.492	-1.019	0.490
Pos._F	-0.4428	0.300	-1.476	0.141	-1.033	0.147
Pos._F/C	0.2234	0.825	0.271	0.787	-1.399	1.846
Pos._G/F	0.5504	0.812	0.678	0.498	-1.046	2.147

Omnibus:	23.831	Durbin-Watson:	1.427
Prob(Omnibus):	0.000	Jarque-Bera (JB):	71.792
Skew:	0.161	Prob(JB):	2.57e-16
Kurtosis:	5.122	Cond. No.	2.47e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.47e+03. This might indicate that there are strong multicollinearity or other numerical problems.
OSR2: 0.7835478120135656

[26]:
Height	2.419713
GP	1.634408
MIN	5.295664
FG%	1.420990
3PT%	1.535685
FT%	1.512790

```
OREB        3.791196
DREB        4.502331
AST         3.032762
STL         2.184249
BLK         2.616033
Pos._F      1.343541
Pos._F/C    1.136659
Pos._G/F    1.100756
dtype: float64
```

[27]:
```
#MIN - removed / had the next largest VIF higher than 5
cols = ['Height', 'GP', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL', 'BLK','Pos._F', 'Pos._F/C',
       'Pos._G/F']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

X_ = sm.add_constant(X_)
X_t = sm.add_constant(X_t)

model_4 = sm.OLS(y_, X_).fit()
print(model_4.summary())
print('OSR2:', OSR2(model_4, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    PTS   R-squared:                       0.754
Model:                            OLS   Adj. R-squared:                  0.745
Method:                 Least Squares   F-statistic:                     84.79
Date:                Fri, 02 Jul 2021   Prob (F-statistic):          5.57e-101
Time:                        21:26:31   Log-Likelihood:                -874.59
No. Observations:                 374   AIC:                             1777.
Df Residuals:                     360   BIC:                             1832.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3112      3.845      0.081      0.936      -7.250       7.872
Height        -0.0251      0.051     -0.492      0.623      -0.125       0.075
GP            -0.0304      0.014     -2.118      0.035      -0.059      -0.002
FG%            1.5823      0.954      1.659      0.098      -0.294       3.458
3PT%           4.2498      0.817      5.203      0.000       2.644       5.856
```

```
FT%            2.0253      0.584      3.470      0.001       0.877       3.173
OREB           1.0390      0.335      3.102      0.002       0.380       1.698
DREB           1.2470      0.165      7.571      0.000       0.923       1.571
AST            1.2766      0.167      7.646      0.000       0.948       1.605
STL            1.5687      0.410      3.824      0.000       0.762       2.375
BLK            0.1851      0.424      0.436      0.663      -0.649       1.020
Pos._F        -0.4847      0.334     -1.449      0.148      -1.143       0.173
Pos._F/C      -0.0205      0.919     -0.022      0.982      -1.828       1.787
Pos._G/F       0.9390      0.904      1.039      0.300      -0.838       2.716
==============================================================================
Omnibus:                       27.545   Durbin-Watson:                   1.719
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               49.007
Skew:                           0.457   Prob(JB):                     2.28e-11
Kurtosis:                       4.519   Cond. No.                     2.37e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.37e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
OSR2: 0.7124083014972704
```

[27]:
```
Height      2.338910
GP          1.614027
FG%         1.419236
3PT%        1.515515
FT%         1.476065
OREB        3.575632
DREB        3.929282
AST         2.446691
STL         1.994383
BLK         2.575486
Pos._F      1.343244
Pos._F/C    1.135535
Pos._G/F    1.097904
dtype: float64
```

[28]:
```
#Pos._F/C - removed / insignificant variable / high p-value
cols = ['Height', 'GP', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL', 'BLK','Pos._F',
       'Pos._G/F']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']
```

```
X_ = sm.add_constant(X_)
X_t = sm.add_constant(X_t)

model_5 = sm.OLS(y_, X_).fit()
print(model_5.summary())
print('OSR2:', OSR2(model_5, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    PTS   R-squared:                       0.754
Model:                            OLS   Adj. R-squared:                  0.746
Method:                 Least Squares   F-statistic:                     92.11
Date:                Fri, 02 Jul 2021   Prob (F-statistic):          5.67e-102
Time:                        21:26:31   Log-Likelihood:                -874.59
No. Observations:                 374   AIC:                             1775.
Df Residuals:                     361   BIC:                             1826.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3296      3.750      0.088      0.930      -7.045       7.704
Height        -0.0253      0.050     -0.510      0.610      -0.123       0.072
GP            -0.0304      0.014     -2.123      0.034      -0.059      -0.002
FG%            1.5830      0.952      1.662      0.097      -0.290       3.455
3PT%           4.2488      0.814      5.217      0.000       2.647       5.851
FT%            2.0251      0.583      3.475      0.001       0.879       3.171
OREB           1.0392      0.334      3.109      0.002       0.382       1.697
DREB           1.2470      0.164      7.581      0.000       0.924       1.570
AST            1.2767      0.167      7.663      0.000       0.949       1.604
STL            1.5687      0.410      3.829      0.000       0.763       2.374
BLK            0.1845      0.423      0.436      0.663      -0.647       1.016
Pos._F        -0.4832      0.327     -1.480      0.140      -1.125       0.159
Pos._G/F       0.9406      0.900      1.045      0.297      -0.829       2.710
==============================================================================
Omnibus:                       27.574   Durbin-Watson:                   1.719
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               49.080
Skew:                           0.458   Prob(JB):                     2.20e-11
Kurtosis:                       4.520   Cond. No.                     2.31e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.31e+03. This might indicate that there are
```

```
strong multicollinearity or other numerical problems.
OSR2: 0.7123717895887276
```

[28]:
```
Height      2.227015
GP          1.612177
FG%         1.417933
3PT%        1.511363
FT%         1.475774
OREB        3.571522
DREB        3.929280
AST         2.442583
STL         1.994382
BLK         2.563458
Pos._F      1.283863
Pos._G/F    1.091243
dtype: float64
```

[29]:
```
#const - removed / insignificant variable / high p-value
cols = ['Height', 'GP', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL', 'BLK','Pos._F',
       'Pos._G/F']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

# X_ = sm.add_constant(X_)
# X_t = sm.add_constant(X_t)

model_6 = sm.OLS(y_, X_).fit()
print(model_6.summary())
print('OSR2:', OSR2(model_6, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                     PTS   R-squared (uncentered):
0.901
Model:                             OLS   Adj. R-squared (uncentered):
0.898
Method:                  Least Squares   F-statistic:
274.0
Date:                 Fri, 02 Jul 2021   Prob (F-statistic):
2.33e-173
Time:                         21:26:31   Log-Likelihood:
-874.60
```

```
No. Observations:                 374   AIC:
1773.
Df Residuals:                     362   BIC:
1820.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Height        -0.0210      0.006     -3.468      0.001      -0.033      -0.009
GP            -0.0305      0.014     -2.137      0.033      -0.059      -0.002
FG%            1.5710      0.941      1.669      0.096      -0.280       3.422
3PT%           4.2651      0.792      5.385      0.000       2.708       5.823
FT%            2.0262      0.582      3.482      0.001       0.882       3.170
OREB           1.0403      0.334      3.118      0.002       0.384       1.696
DREB           1.2444      0.162      7.703      0.000       0.927       1.562
AST            1.2803      0.161      7.940      0.000       0.963       1.597
STL            1.5747      0.403      3.904      0.000       0.781       2.368
BLK            0.1735      0.403      0.430      0.667      -0.620       0.967
Pos._F        -0.4897      0.318     -1.542      0.124      -1.114       0.135
Pos._G/F       0.9325      0.894      1.043      0.298      -0.825       2.690
==============================================================================
Omnibus:                       27.475   Durbin-Watson:                   1.719
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               48.851
Skew:                           0.456   Prob(JB):                     2.47e-11
Kurtosis:                       4.517   Cond. No.                         620.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
OSR2: 0.7120481137226101
```

[29]:
```
Height      2.227015
GP          1.612177
FG%         1.417933
3PT%        1.511363
FT%         1.475774
OREB        3.571522
DREB        3.929280
AST         2.442583
STL         1.994382
BLK         2.563458
Pos._F      1.283863
Pos._G/F    1.091243
dtype: float64
```

[30]:
```
#BLK - removed / insignificant variable / high p-value
cols = ['Height', 'GP', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL','Pos._F',
       'Pos._G/F']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

model_7 = sm.OLS(y_, X_).fit()
print(model_7.summary())
print('OSR2:', OSR2(model_7, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                      PTS   R-squared (uncentered):
0.901
Model:                              OLS   Adj. R-squared (uncentered):
0.898
Method:                   Least Squares   F-statistic:
299.6
Date:                  Fri, 02 Jul 2021   Prob (F-statistic):
1.44e-174
Time:                          21:26:31   Log-Likelihood:
-874.69
No. Observations:                   374   AIC:
1771.
Df Residuals:                       363   BIC:
1815.
Df Model:                            11
Covariance Type:             nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Height        -0.0210      0.006     -3.474      0.001      -0.033      -0.009
GP            -0.0305      0.014     -2.136      0.033      -0.059      -0.002
FG%            1.6240      0.932      1.742      0.082      -0.209       3.457
3PT%           4.1949      0.774      5.419      0.000       2.672       5.717
FT%            2.0279      0.581      3.489      0.001       0.885       3.171
OREB           1.0968      0.306      3.581      0.000       0.494       1.699
DREB           1.2586      0.158      7.969      0.000       0.948       1.569
AST            1.2706      0.159      7.967      0.000       0.957       1.584
STL            1.5651      0.402      3.890      0.000       0.774       2.356
Pos._F        -0.5055      0.315     -1.604      0.110      -1.125       0.114
```

```
Pos._G/F       0.9168      0.892      1.028      0.305      -0.838       2.671
==============================================================================
Omnibus:                       26.906   Durbin-Watson:                   1.717
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               46.882
Skew:                           0.455   Prob(JB):                     6.60e-11
Kurtosis:                       4.477   Cond. No.                         611.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
OSR2: 0.7138796465737988
```

[30]:
```
Height      2.036281
GP          1.611917
FG%         1.408506
3PT%        1.487866
FT%         1.475529
OREB        3.038136
DREB        3.853361
AST         2.434689
STL         1.994363
Pos._F      1.244671
Pos._G/F    1.086042
dtype: float64
```

[31]:
```
#Pos._G/F - removed / insignificant variable / high p-value
cols = ['Height', 'GP', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL','Pos._F']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

model_8 = sm.OLS(y_, X_).fit()
print(model_8.summary())
print('OSR2:', OSR2(model_8, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                                 OLS Regression Results
==============================================================================
=======
Dep. Variable:                    PTS   R-squared (uncentered):
0.900
Model:                            OLS   Adj. R-squared (uncentered):
0.898
Method:                 Least Squares   F-statistic:
```

```
329.4
Date:                Fri, 02 Jul 2021   Prob (F-statistic):
1.31e-175
Time:                        21:26:31   Log-Likelihood:
-875.24
No. Observations:                 374   AIC:
1770.
Df Residuals:                     364   BIC:
1810.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Height        -0.0207      0.006     -3.432      0.001      -0.033      -0.009
GP            -0.0314      0.014     -2.209      0.028      -0.059      -0.003
FG%            1.5694      0.931      1.687      0.093      -0.261       3.399
3PT%           4.2813      0.770      5.563      0.000       2.768       5.795
FT%            2.0489      0.581      3.527      0.000       0.907       3.191
OREB           1.0757      0.306      3.520      0.000       0.475       1.677
DREB           1.2893      0.155      8.313      0.000       0.984       1.594
AST            1.2565      0.159      7.907      0.000       0.944       1.569
STL            1.5466      0.402      3.848      0.000       0.756       2.337
Pos._F        -0.5413      0.313     -1.728      0.085      -1.157       0.075
==============================================================================
Omnibus:                       28.341   Durbin-Watson:                   1.712
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               48.842
Skew:                           0.481   Prob(JB):                     2.48e-11
Kurtosis:                       4.486   Cond. No.                         600.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
OSR2: 0.7136047628872091
```

[31]:
```
Height    2.019395
GP        1.603572
FG%       1.401402
3PT%      1.463477
FT%       1.473525
OREB      3.021423
DREB      3.751372
AST       2.426246
STL       1.992717
Pos._F    1.225470
dtype: float64
```

[32]:
```
#FG% - removed / insignificant variable / high p-value
cols = ['Height', 'GP', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL','Pos._F']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

model_9 = sm.OLS(y_, X_).fit()
print(model_9.summary())
print('OSR2:', OSR2(model_9, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                    PTS   R-squared (uncentered):
0.900
Model:                            OLS   Adj. R-squared (uncentered):
0.897
Method:                 Least Squares   F-statistic:
363.8
Date:                Fri, 02 Jul 2021   Prob (F-statistic):
2.74e-176
Time:                        21:26:31   Log-Likelihood:
-876.69
No. Observations:                 374   AIC:
1771.
Df Residuals:                     365   BIC:
1807.
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Height        -0.0147      0.005     -3.010      0.003      -0.024      -0.005
GP            -0.0297      0.014     -2.086      0.038      -0.058      -0.002
3PT%           4.5784      0.751      6.096      0.000       3.101       6.055
FT%            2.0256      0.582      3.479      0.001       0.881       3.170
OREB           1.1446      0.304      3.770      0.000       0.548       1.742
DREB           1.3268      0.154      8.623      0.000       1.024       1.629
AST            1.2667      0.159      7.957      0.000       0.954       1.580
STL            1.4666      0.400      3.665      0.000       0.680       2.253
Pos._F        -0.5706      0.314     -1.820      0.070      -1.187       0.046
================================================================================
Omnibus:                       25.366   Durbin-Watson:                   1.750
```

```
Prob(Omnibus):                  0.000   Jarque-Bera (JB):                42.657
Skew:                           0.443   Prob(JB):                     5.46e-10
Kurtosis:                       4.398   Cond. No.                         473.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
OSR2: 0.7128759283386841
```

[32]:
```
Height    1.900183
GP        1.597674
3PT%      1.362100
FT%       1.472892
OREB      2.979945
DREB      3.718413
AST       2.409117
STL       1.978626
Pos._F    1.216786
dtype: float64
```

[33]:
```
#Pos._F - removed / insignificant variable / high p-value
cols = ['Height', 'GP', '3PT%', 'FT%', 'OREB', 'DREB',
       'AST', 'STL']
X_ = basketballtrain_dumm[cols]
y_ = basketballtrain_dumm['PTS']

X_t = basketballtest_dumm[cols]
y_t = basketballtest_dumm['PTS']

model_10 = sm.OLS(y_, X_).fit()
print(model_10.summary())
print('OSR2:', OSR2(model_10, X_t, y_t, y_))
VIF(basketballtrain_dumm, cols)
```

```
                                 OLS Regression Results
==============================================================================
=======
Dep. Variable:                    PTS   R-squared (uncentered):
0.899
Model:                            OLS   Adj. R-squared (uncentered):
0.897
Method:                 Least Squares   F-statistic:
406.3
Date:                Fri, 02 Jul 2021   Prob (F-statistic):
6.81e-177
Time:                        21:26:31   Log-Likelihood:
-878.38
```

```
No. Observations:                 374   AIC:
1773.
Df Residuals:                     366   BIC:
1804.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Height        -0.0173      0.005     -3.684      0.000      -0.027      -0.008
GP            -0.0312      0.014     -2.188      0.029      -0.059      -0.003
3PT%           4.7093      0.750      6.279      0.000       3.234       6.184
FT%            2.0492      0.584      3.510      0.001       0.901       3.197
OREB           1.0788      0.302      3.567      0.000       0.484       1.673
DREB           1.3264      0.154      8.593      0.000       1.023       1.630
AST            1.3115      0.158      8.313      0.000       1.001       1.622
STL            1.4954      0.401      3.729      0.000       0.707       2.284
==============================================================================
Omnibus:                       30.485   Durbin-Watson:                   1.780
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.950
Skew:                           0.504   Prob(JB):                     1.93e-12
Kurtosis:                       4.564   Cond. No.                         471.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
OSR2: 0.7116513368154673
```

[33]:
```
Height    1.827124
GP        1.594566
3PT%      1.358409
FT%       1.472384
OREB      2.950555
DREB      3.711103
AST       2.384387
STL       1.978565
dtype: float64
```

5 2. Decision Tree Regressor

[34]:
```
from sklearn.preprocessing import OneHotEncoder
drop_enc = OneHotEncoder(drop='first').fit(basketballtrain[['Pos.']])
print(drop_enc.categories_)

# Perform the transformation for both the training and the test set.
```

```
X_train_categorical = drop_enc.transform(basketballtrain[['Pos.']]).toarray()
X_train_numerical = X_train[['GP', 'MIN', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',␣
↪'REB', 'AST',
       'STL', 'BLK', 'Height']].values

# combine the numerical variables and the one-hot encoded categorical variables
basketballtrain_enc = np.concatenate((X_train_numerical,X_train_categorical),␣
↪axis = 1)

X_test_categorical = drop_enc.transform(basketballtest[['Pos.']]).toarray()
X_test_numerical = X_test[['GP', 'MIN', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB',␣
↪'REB', 'AST',
       'STL', 'BLK', 'Height']].values
basketballtest_enc = np.concatenate((X_test_numerical,X_test_categorical), axis␣
↪= 1)
columns = ['GP', 'MIN', 'FG%', '3PT%', 'FT%', 'OREB', 'DREB', 'REB', 'AST',
       'STL', 'BLK', 'Height', 'F', 'F/C', 'G', 'G/F']
```

```
[array(['C', 'F', 'F/C', 'G', 'G/F'], dtype=object)]
```

[35]:
```
from sklearn.model_selection import GridSearchCV

grid_values = {'ccp_alpha': np.linspace(0, 2, 500),
               'min_samples_leaf': [5],
               'min_samples_split': [20],
               'max_depth': [30],
               'random_state': [88]}

dtr = DecisionTreeRegressor()
cv = KFold(n_splits=10,random_state=333,shuffle=True)
dtr_cv = GridSearchCV(dtr, param_grid = grid_values, scoring = 'r2', cv=cv,␣
↪verbose=1)
dtr_cv.fit(basketballtrain_enc, y_train)
```

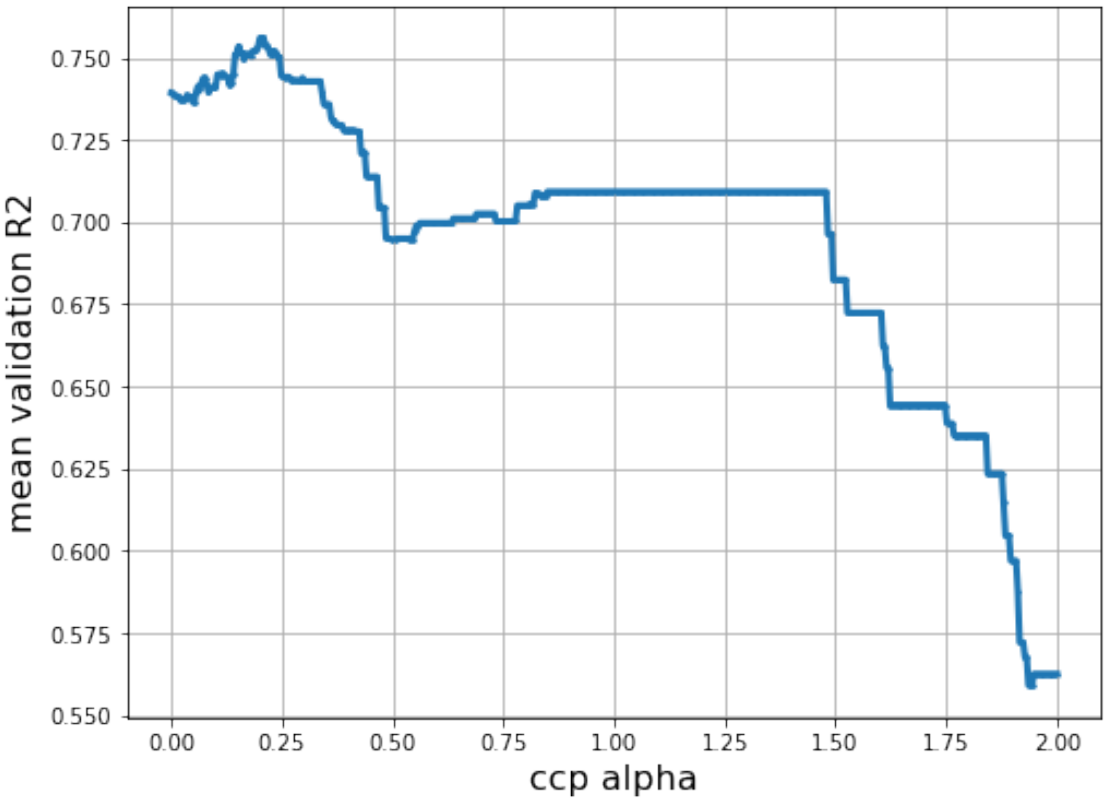
```
Fitting 10 folds for each of 500 candidates, totalling 5000 fits
```

[35]:
```
GridSearchCV(cv=KFold(n_splits=10, random_state=333, shuffle=True),
             estimator=DecisionTreeRegressor(),
             param_grid={'ccp_alpha': array([0.        , 0.00400802, 0.00801603,
0.01202405, 0.01603206,
       0.02004008, 0.0240481 , 0.02805611, 0.03206413, 0.03607214,
       0.04008016, 0.04408818, 0.04809619, 0.05210421, 0.05611222,
       0.06012024, 0.06412826, 0.06813627, 0.07214429, 0.0761523 ,
       0.08016032…
       1.90380762, 1.90781563, 1.91182365, 1.91583166, 1.91983968,
       1.9238477 , 1.92785571, 1.93186373, 1.93587174, 1.93987976,
       1.94388778, 1.94789579, 1.95190381, 1.95591182, 1.95991984,
```

```
        1.96392786, 1.96793587, 1.97194389, 1.9759519 , 1.97995992,
        1.98396794, 1.98797595, 1.99198397, 1.99599198, 2.        ]),
                     'max_depth': [30], 'min_samples_leaf': [5],
                     'min_samples_split': [20], 'random_state': [88]},
             scoring='r2', verbose=1)
```

[36]:
```
acc = dtr_cv.cv_results_['mean_test_score']
ccp = dtr_cv.cv_results_['param_ccp_alpha'].data
```

[37]:
```
#Plot to see which ccp_alpha has a higher validation R2
plt.figure(figsize=(8, 6))
plt.xlabel('ccp alpha', fontsize=16)
plt.ylabel('mean validation R2', fontsize=16)
plt.scatter(ccp, acc, s=2)
plt.plot(ccp, acc, linewidth=3)
plt.grid(True, which='both')
plt.show()
print('Grid best parameter ccp_alpha (max. R2): ', dtr_cv.
 ↪best_params_['ccp_alpha'])
print('Grid best score (R2): ', dtr_cv.best_score_)
```



```
Grid best parameter ccp_alpha (max. R2):  0.2004008016032064
Grid best score (R2):  0.7560936276601745
```

[38]:
```
#Below is the code for building a decision tree regressor model with ccp_alpha
 ↪= 0.2004008
dtr2 = DecisionTreeRegressor(min_samples_split=10,
                             ccp_alpha=0.2004008,
                             random_state = 88)
dtr2 = dtr2.fit(basketballtrain_enc, y_train)
dtr2
```

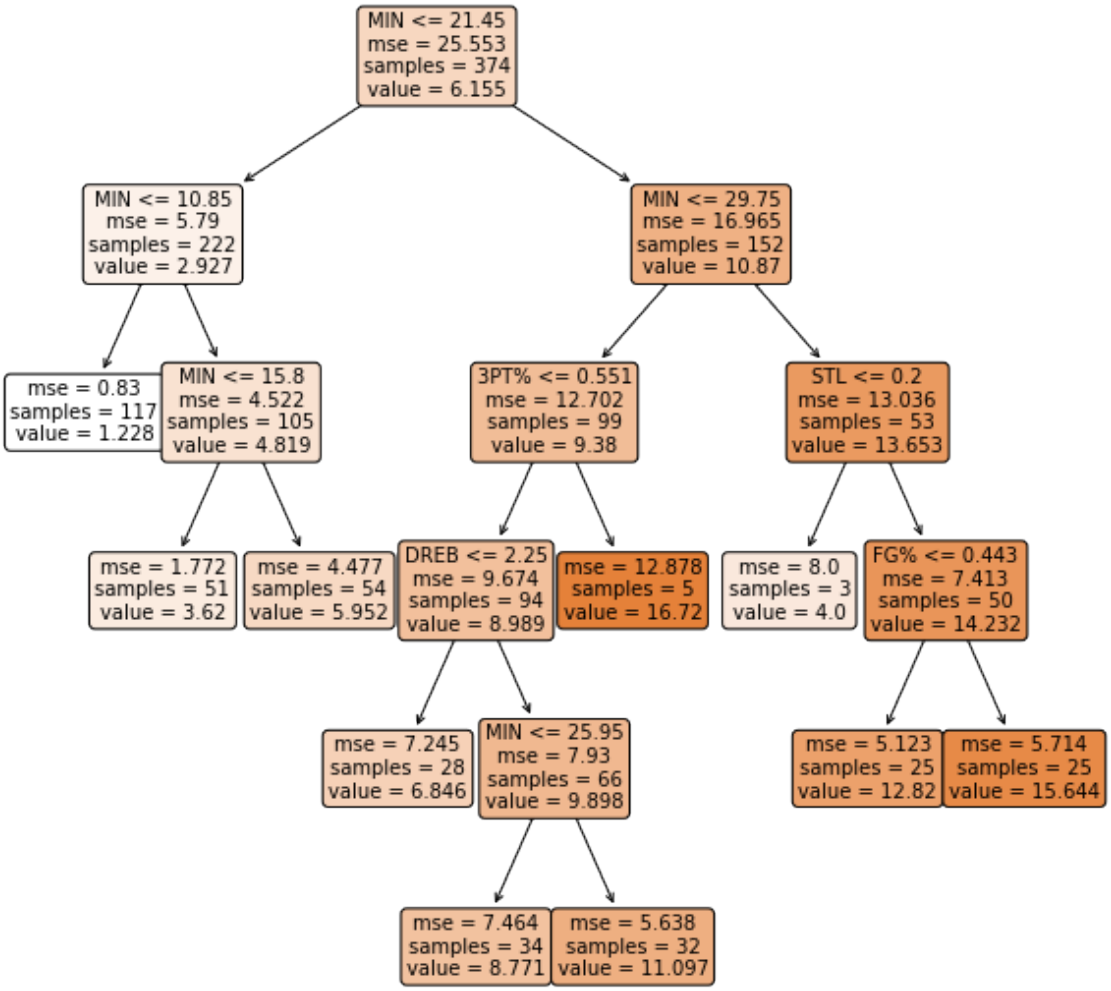
[38]:
```
DecisionTreeRegressor(ccp_alpha=0.2004008, min_samples_split=10,
                      random_state=88)
```

[39]:
```
print('Decision Tree Regressor OSR2:', OSR2(dtr2, basketballtest_enc, y_test,
 ↪y_train))
```

```
Decision Tree Regressor OSR2: 0.7521723750941789
```

[40]:
```
print('Node count =', dtr2.tree_.node_count)
plt.figure(figsize=(10,10))
plot_tree(dtr2,
          feature_names=columns,
          class_names=['0','1'],
          filled=True,
          impurity=True,
          rounded=True,
          fontsize=10)
plt.show()
```

```
Node count = 19
```



[41]:
```
pd.DataFrame({'Feature' : columns,
              'Importance score': 100 * dtr2.feature_importances_}).round(1).
↪sort_values('Importance score', ascending=False)
```

[41]:

	Feature	Importance score
1	MIN	89.4
9	STL	3.6
3	3PT%	3.5
6	DREB	2.3
2	FG%	1.2
0	GP	0.0
4	FT%	0.0
5	OREB	0.0
7	REB	0.0

```
8       AST              0.0
10      BLK              0.0
11   Height              0.0
12        F              0.0
13      F/C              0.0
14        G              0.0
15      G/F              0.0
```

6 3. Random Forest Regressor

[42]:
```
from sklearn.model_selection import GridSearchCV

grid_values = {'n_estimators': np.arange(1, 100, 10),
               'max_features': np.linspace(1, 18, 18),
               'min_samples_leaf': [5],
               'min_samples_split': [20],
               'random_state': [88]}

rfr = RandomForestRegressor()
cv = KFold(n_splits=10,random_state=333,shuffle=True)
rfr_cv = GridSearchCV(rfr, param_grid = grid_values, scoring = 'r2', cv=cv)
rfr_cv.fit(basketballtrain_enc, y_train)
```

[42]:
```
GridSearchCV(cv=KFold(n_splits=10, random_state=333, shuffle=True),
             estimator=RandomForestRegressor(),
             param_grid={'max_features': array([ 1.,  2.,  3.,  4.,  5.,  6.,
7.,  8.,  9., 10., 11., 12., 13.,
       14., 15., 16., 17., 18.]),
                         'min_samples_leaf': [5], 'min_samples_split': [20],
                         'n_estimators': array([ 1, 11, 21, 31, 41, 51, 61, 71,
81, 91]),
                         'random_state': [88]},
             scoring='r2')
```

[43]:
```
print('Best n_estimators:', rfr_cv.best_params_['n_estimators'])
print('Best max_features:', rfr_cv.best_params_['max_features'])
```

```
Best n_estimators: 81
Best max_features: 1.0
```

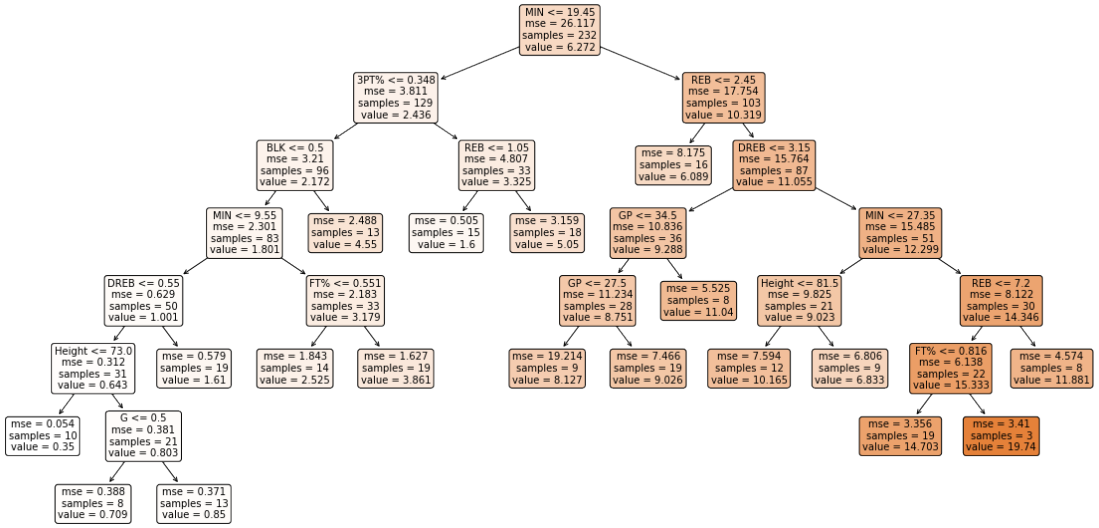
[44]:
```
#Below is the code for building a random forest regressor model
#with n_estimators=81, max_features=1, min_samples_split=20
rfr2 = RandomForestRegressor(n_estimators=81,
                             max_features= 1,
                             min_samples_split=20,
                            random_state = 88)
```

```
rfr2 = rfr2.fit(basketballtrain_enc, y_train)
rfr2


print('Random Forest Regressor OSR2:', OSR2(rfr2, basketballtest_enc, y_test,␣
 ↪y_train))
```

```
Random Forest Regressor OSR2: 0.7907608006274502
```

[45]:
```
from sklearn import tree
plt.figure(figsize=(20,10))
_ = tree.plot_tree(rfr2.estimators_[0], feature_names=columns,␣
 ↪filled=True,impurity=True,
          rounded=True,fontsize=10)
```



[46]:
```
pd.DataFrame({'Feature' : columns,
              'Importance score': 100 * rfr2.feature_importances_}).round(1).
 ↪sort_values('Importance score', ascending=False)
```

[46]:

	Feature	Importance score
1	MIN	14.1
8	AST	12.8
6	DREB	11.2
9	STL	9.9
7	REB	9.8
5	OREB	7.7
4	FT%	7.6
2	FG%	6.9
3	3PT%	6.5

```
0       GP              5.0
10     BLK              3.1
11  Height              2.4
12       F              1.0
15     G/F              0.9
14       G              0.8
13     F/C              0.3
```

7 4. Ridge Regression

[66]:
```
X_train
```

[66]:
```
      GP   MIN    FG%   3PT%    FT%  OREB  DREB  REB  AST  STL  BLK  Height
0   31.0  34.5  0.432  0.302  0.782   1.5   4.5  6.0  2.0  1.7  0.5    79.0
1   29.0  34.4  0.440  0.281  0.727   0.3   2.8  3.1  4.3  1.4  0.2    72.0
2   31.0  32.6  0.391  0.349  0.696   0.2   1.2  1.4  0.9  1.5  0.1    75.0
3   31.0  28.4  0.411  0.472  0.791   0.9   2.7  3.6  2.0  0.9  0.4    76.0
4   28.0  17.5  0.469  0.355  0.667   0.8   2.2  3.0  0.2  0.3  1.3    87.0
..   …   …    …    …    …    …   …   …  …  …  …  …      …
8   20.0   7.9  0.531  0.000  0.400   0.5   0.9  1.4  0.4  0.4  0.4    80.0
9    4.0   2.5  0.333  0.286  0.000   0.0   0.8  0.8  0.0  0.3  0.0    77.0
10  12.0   6.3  0.364  0.000  0.800   0.5   0.8  1.3  0.1  0.1  0.0    82.0
11  24.0   8.8  0.421  0.250  0.429   0.2   0.5  0.6  0.5  0.3  0.0    74.0
12   2.0   1.5  0.000  0.000  0.000   0.5   0.5  1.0  0.5  0.5  0.0    80.0

[374 rows x 12 columns]
```

[65]:
```
y_train
```

[65]:
```
0     14.3
1     11.6
2     11.0
3     10.8
4      7.5
      …
8      2.0
9      2.0
10     1.0
11     0.9
12     0.0
Name: PTS, Length: 374, dtype: float64
```

[69]:
```
X_test = X_test.drop(['Pos.'],axis=1)
X_test
```

[69]:
```
       GP   MIN    FG%   3PT%    FT%  OREB  DREB  REB  AST  STL  BLK  Height
0    22.0  30.2  0.455  0.364  0.821   1.0   3.6  4.6  1.7  0.4  0.2    76.0
1    29.0  25.9  0.587  0.000  0.590   1.8   4.7  6.4  0.6  0.6  0.5    80.0
2    25.0  29.5  0.408  0.366  0.857   0.8   3.8  4.5  1.8  0.5  0.4    80.0
3    29.0  27.3  0.354  0.327  0.800   0.3   3.0  3.4  1.2  0.3  0.3    77.0
4    29.0  20.5  0.332  0.317  0.868   0.2   1.0  1.2  1.6  0.5  0.0    73.0
..    …    …     …      …      …     …   …   …   …   …   …      …
7    32.0  19.8  0.448  0.286  0.731   1.3   2.8  4.1  0.9  0.6  0.3    79.0
8    22.0  16.5  0.415  0.390  0.900   0.5   2.0  2.5  0.7  0.5  0.1    76.0
9     5.0   4.0  0.500  0.571  0.000   0.4   1.0  1.4  0.2  0.0  0.2    75.0
10   10.0   4.4  0.500  0.250  0.000   0.1   0.6  0.7  0.2  0.1  0.0    78.0
11   29.0   9.0  0.500  0.167  1.000   0.1   0.6  0.7  0.4  0.1  0.3    76.0

[141 rows x 12 columns]
```

[70]:
```
y_test
```

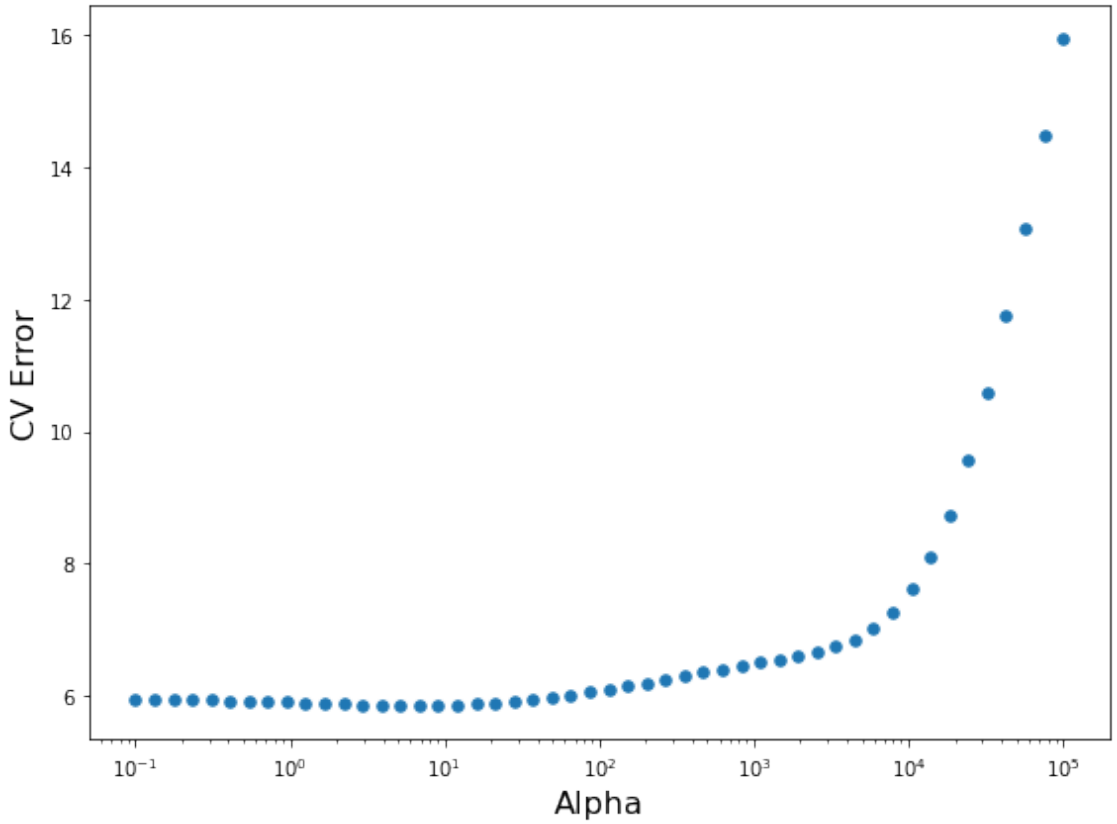
[70]:
```
0     18.0
1     10.3
2      8.9
3      8.5
4      7.2
      …
7      5.8
8      3.6
9      2.4
10     1.5
11     1.0
Name: PTS, Length: 141, dtype: float64
```

[71]:
```
alpha_grid = {'alpha': np.logspace(-1, 5, num=50, base=10)}
rr = Ridge(random_state=88)
rr_cv = GridSearchCV(rr, alpha_grid, scoring='neg_mean_squared_error', cv=10)
rr_cv.fit(X_train, y_train)
```

[71]:
```
GridSearchCV(cv=10, estimator=Ridge(random_state=88),
             param_grid={'alpha': array([1.00000000e-01, 1.32571137e-01,
1.75751062e-01, 2.32995181e-01,
       3.08884360e-01, 4.09491506e-01, 5.42867544e-01, 7.19685673e-01,
       9.54095476e-01, 1.26485522e+00, 1.67683294e+00, 2.22299648e+00,
       2.94705170e+00, 3.90693994e+00, 5.17947468e+00, 6.86648845e+00,
       9.10298178e+00, 1.20679264e+01, 1…
       2.68269580e+02, 3.55648031e+02, 4.71486636e+02, 6.25055193e+02,
       8.28642773e+02, 1.09854114e+03, 1.45634848e+03, 1.93069773e+03,
       2.55954792e+03, 3.39322177e+03, 4.49843267e+03, 5.96362332e+03,
       7.90604321e+03, 1.04811313e+04, 1.38949549e+04, 1.84206997e+04,
       2.44205309e+04, 3.23745754e+04, 4.29193426e+04, 5.68986603e+04,
```

```
       7.54312006e+04, 1.00000000e+05])},
             scoring='neg_mean_squared_error')
```

[72]:
```
range_alpha = rr_cv.cv_results_['param_alpha'].data
CV_scores = rr_cv.cv_results_['mean_test_score']*(-1)
plt.figure(figsize=(8, 6))
ax = plt.gca()
ax.set_xscale('log')
plt.xlabel('Alpha', fontsize=16)
plt.ylabel('CV Error', fontsize=16)
plt.scatter(range_alpha, CV_scores, s=30)
plt.tight_layout()
plt.show()
```



[73]:
```
print(rr_cv.best_params_)
```

```
{'alpha': 6.8664884500430015}
```

[76]:
```
real_ridge = Ridge(alpha=6.8664884500430015, fit_intercept=True)
real_ridge.fit(X_train,y_train)
ridge_osr2=OSR2(real_ridge, X_test, y_test, y_train)
```

```
rpredict_train=real_ridge.predict(X_train)
rpredict_test=real_ridge.predict(X_test)

ridge_train_mse=mean_squared_error(y_train, rpredict_train)
ridge_test_mse=mean_squared_error(y_test, rpredict_test)

ridge_train_mae=mean_absolute_error(y_train, rpredict_train)
ridge_test_mae=mean_absolute_error(y_test, rpredict_test)

print(ridge_osr2)
print(ridge_train_mse, ridge_test_mse)
print(ridge_train_mae, ridge_test_mae)
```

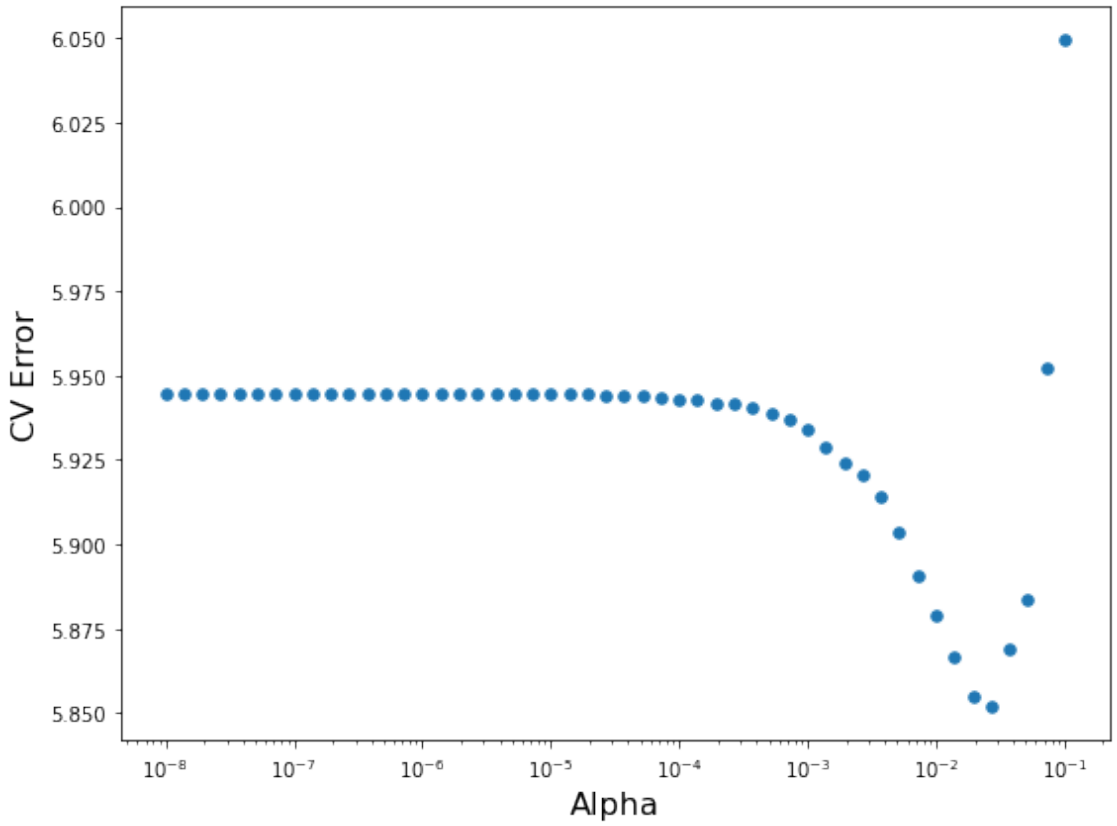
```
0.7875554940275479
5.162796775173004 4.8647049051654365
1.6725254120444795 1.6083931806110583
```

8 5. Lasso Regression

[77]:
```
alphas = np.logspace(-8, 1 , num=50, base=10)

for a in alphas:
    lasso = Lasso(alpha=a, random_state=88)

alpha_grid = {'alpha': np.logspace(-8, -1, num=50, base=10)}
lasso_cv = GridSearchCV(lasso, alpha_grid, scoring='neg_mean_squared_error',␣
 ↪cv=10)
lasso_cv.fit(X_train, y_train)
range_alpha = lasso_cv.cv_results_['param_alpha'].data
CV_scores = lasso_cv.cv_results_['mean_test_score']*(-1)
plt.figure(figsize=(8, 6))
ax = plt.gca()
ax.set_xscale('log')
plt.xlabel('Alpha', fontsize=16)
plt.ylabel('CV Error', fontsize=16)
plt.scatter(range_alpha, CV_scores, s=30)
plt.tight_layout()
plt.show()
```



[78]:
```
print(lasso_cv.best_params_)
```

```
{'alpha': 0.026826957952797218}
```

[80]:
```
real_lasso = Lasso(alpha=0.026826957952797218, fit_intercept=True)
real_lasso.fit(X_train,y_train)
lasso_osr2=OSR2(real_lasso, X_test, y_test, y_train)

lpredict_train=real_lasso.predict(X_train)
lpredict_test=real_lasso.predict(X_test)

lasso_train_mse=mean_squared_error(y_train, lpredict_train)
lasso_test_mse=mean_squared_error(y_test, lpredict_test)

lasso_train_mae=mean_absolute_error(y_train, lpredict_train)
lasso_test_mae=mean_absolute_error(y_test, lpredict_test)

print(lasso_osr2)
print(lasso_train_mse, lasso_test_mse)
print(lasso_train_mae, lasso_test_mae)
```

```
0.7861275451897343
5.158138118416417 4.897403089963579
1.6768396057603625 1.6188829359848993
```

9 6. Final Comparison Table

[83]:
```
#Creating comparison Table
comparison_data = {'Linear Regression': ['{:.3f}'.format(OSR2(model_10, X_t,␣
↪y_t, y_)),
                                         '{:.4f}'.
↪format(mean_squared_error(y_t, model_10.predict(X_t))),
                                         '{:.3f}'.
↪format(mean_absolute_error(y_t, model_10.predict(X_t)))],
                   'Decision Tree Regressor': ['{:.3f}'.format(OSR2(dtr2,␣
↪basketballtest_enc, y_test, y_train)),
                                               '{:.4f}'.
↪format(mean_squared_error(y_test, dtr2.predict(basketballtest_enc))),
                                         '{:.3f}'.
↪format(mean_absolute_error(y_test, dtr2.predict(basketballtest_enc)))],
                   'Lasso Regression' : ['{:.3f}'.format(lasso_osr2),
                                         '{:.4f}'.format((lasso_test_mse)),
                                         '{:.3f}'.format(lasso_test_mae)],
                   'Ridge Regression' : ['{:.3f}'.format(ridge_osr2),
                                         '{:.4f}'.format((ridge_test_mse)),
                                         '{:.3f}'.format(ridge_test_mae)],
                   'Random Forest Regressor': ['{:.3f}'.format(OSR2(rfr2,␣
↪basketballtest_enc, y_test, y_train)),
                                               '{:.4f}'.
↪format(mean_squared_error(y_test, rfr2.predict(basketballtest_enc))),
                                         '{:.3f}'.
↪format(mean_absolute_error(y_test, rfr2.predict(basketballtest_enc)))],
}

comparison_table = pd.DataFrame(data=comparison_data, index=['OS R-squared',␣
↪'Out-of-sample MSE', 'Out-of-sample MAE'])
comparison_table
```

[83]:

	Linear Regression	Decision Tree Regressor	Lasso Regression	\
OS R-squared	0.712	0.752	0.786	
Out-of-sample MSE	6.6028	5.6749	4.8974	
Out-of-sample MAE	1.861	1.783	1.619	

	Ridge Regression	Random Forest Regressor
OS R-squared	0.788	0.791
Out-of-sample MSE	4.8647	4.7913
Out-of-sample MAE	1.608	1.652

10 7. Prediction on scores using Random Forest Regressor

[84]:
```
y_pred0 = rfr2.predict(basketballtrain_enc)
y_pred1 = rfr2.predict(basketballtest_enc)
y_pred0 = y_pred0.astype(int)
y_pred1 = y_pred1.astype(int)
y_pred = np.concatenate((y_pred0, y_pred1))
y_pred
```

[84]:
```
array([12, 11,  7, 11,  6,  7,  4,  3,  2,  1,  1,  0,  0,  0,  8, 11, 10,
       10,  6,  8,  4,  2,  1,  1,  1,  1,  1,  1,  0, 12, 11,  9, 10,  5,
        7,  3,  6,  2,  3,  2,  1,  1,  0,  0, 12, 11, 10, 10,  9,  5,  5,
        2,  3,  1,  2,  1,  1,  0,  9,  6, 12, 10,  5,  5,  5,  4,  8, 11,
        3, 10, 10,  9,  8,  3,  6,  6,  6,  6,  4,  4,  1,  1,  0,  0,  2,
        2,  1,  1,  1, 12, 12, 13, 11,  9,  6,  8,  4,  6,  4,  2,  1,  2,
       12, 13, 10,  7,  7,  7,  3,  4,  5,  2,  1,  1,  0, 12,  9, 10,  6,
        5,  9,  5,  4,  2,  3,  4,  1,  2,  8,  9,  8,  6,  5,  7,  8,  6,
        8,  6,  7,  1,  2,  2,  2, 11, 10,  8,  7,  6,  5,  5,  7,  2,  2,
        3,  1, 11, 11,  8,  8,  6,  4,  7,  3,  5,  2,  3,  4,  2,  2, 11,
        8, 12,  7,  6,  5,  7,  5,  4,  3,  2,  1,  0, 10,  9,  7, 11,  9,
        6,  6,  7,  4,  5,  3,  2,  0,  1,  9,  9,  9, 11, 10,  7, 10, 10,
        6,  7,  6,  5,  5,  3,  5,  4,  3,  0,  4,  6,  3,  2,  3,  2,  4,
        1,  2, 12,  8,  8, 10,  9,  9,  5,  3,  3,  3,  3,  2,  2, 12, 13,
        8,  8,  7,  7,  5,  5,  7,  3,  0,  0,  0, 13, 12, 12, 10,  8,  8,
        6,  3,  2,  1,  2,  1,  1, 12, 10, 14, 12, 11, 10,  5,  5,  2,  1,
        1,  1,  1, 13, 12,  8,  9, 11,  9,  3,  3,  2,  1,  3,  0,  0, 13,
        9,  8,  5,  8,  7,  5,  6,  4,  3,  2,  1,  1, 12, 13, 11,  7,  5,
        9,  7,  3,  4,  2,  2, 12, 10,  9,  8,  8,  7,  4,  5,  2,  1,  1,
        1, 11, 10,  9,  9,  7,  9,  7,  2,  2,  1,  1, 15, 10, 10,  7,  7,
        7,  6,  7,  5, 10,  4,  3,  5,  4,  1,  1,  1, 13, 11,  8,  5,  7,
        4,  7,  2,  2, 14, 11, 11,  9,  8,  6,  6,  3,  2,  1,  2,  1,  1,
       11,  8, 10,  7,  4,  3,  7,  5,  4,  2,  2,  2,  1,  1,  0, 12,  8,
        6,  9,  7,  4,  4,  3,  3,  2,  2,  2,  0,  0,  0,  0,  7, 10,  8,
        5, 10, 10,  6,  2,  2,  3,  2,  2,  2,  2, 12, 12,  9,  8,  6, 10,
       12,  2,  2,  2,  1,  3,  1,  1,  1, 10,  7,  9,  5,  5,  5,  7,  5,
        4,  3,  4,  2,  2,  2,  2,  5,  8, 10, 10, 10,  7,  9,  5,  6,  3,
        1,  1, 12, 11, 12,  8, 12,  9,  7,  5,  4,  3,  4,  1,  0,  0, 12,
        9, 10,  6,  7,  5,  5,  4,  5,  2,  1,  1,  0,  8, 12, 12,  8,  6,
        9,  6,  6,  4,  3,  1,  1,  1,  2,  2,  9, 11,  8, 11,  5,  6,  7,
        7,  6,  2,  1,  3])
```

[85]:
```
# Make a dataframe and convert from float to int
predicted_dataset = pd.DataFrame()
predicted_dataset['PTS'] = y_test
predicted_dataset['PRED PTS'] = y_pred1
predicted_dataset
```

[85]:

	PTS	PRED PTS
0	18.0	11
1	10.3	8
2	8.9	10
3	8.5	7
4	7.2	4
..	…	…
7	5.8	7
8	3.6	6
9	2.4	2
10	1.5	1
11	1.0	3

[141 rows x 2 columns]