```
import pandas as pd
import numpy as np
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

draft_data = pd.read_csv("NBADraft.csv")

draft_data.shape

(415, 29)

draft_data.head()

   Pk        School  G_college  GS  MP_college   FG   FGA  FG%_college
2P  \
0   1      Kentucky         37  37        34.8  5.5  11.8        0.461
4.5
1   2    Ohio State        101  94        32.8  5.3  10.6        0.502
4.8
2   3  Georgia Tech         36  35        27.5  5.0   8.1        0.611
5.0
3   5      Kentucky         38  37        23.5  5.4   9.7        0.558
5.4
4   7    Georgetown         65  65        32.6  5.3   9.8        0.543
5.2

   2PA  ...  TRB_college  AST_college  STL  BLK  TOV   PF  PTS_college
SOS  \
0  8.8  ...          4.3          6.5  1.8  0.5  4.0  1.9         16.6
6.82
1  9.1  ...          6.8          4.1  1.6  0.7  3.5  2.7         15.0
7.86
2  8.1  ...          8.4          1.0  0.9  2.1  2.5  2.6         12.4
9.02
3  9.6  ...          9.8          1.0  1.0  1.8  2.1  3.2         15.1
6.82
4  9.3  ...          8.2          3.2  1.5  1.5  2.9  2.5         14.5
9.26

               name  lottery
0         john wall     True
1       evan turner     True
2    derrick favors     True
3  demarcus cousins     True
```

```
4       greg monroe     True

[5 rows x 29 columns]
```

```
draft_data.columns
```

```
Index(['Pk', 'School', 'G_college', 'GS', 'MP_college', 'FG', 'FGA',
       'FG%_college', '2P', '2PA', '2P%', '3P', '3PA', '3P%_college',
'FT',
       'FTA', 'FT%_college', 'ORB', 'DRB', 'TRB_college',
'AST_college', 'STL',
       'BLK', 'TOV', 'PF', 'PTS_college', 'SOS', 'name', 'lottery'],
      dtype='object')
```

```
len(draft_data)
```

```
415
```

```
draft_data.drop(["name","lottery"],axis=1,inplace=True)
```

```
draft_data.isna().sum()
```

```
Pk              0
School          0
G_college       0
GS              0
MP_college      0
FG              0
FGA             0
FG%_college     0
2P              0
2PA             0
2P%             0
3P              0
3PA             0
3P%_college    22
FT              0
FTA             0
FT%_college     0
ORB             0
DRB             0
TRB_college     0
AST_college     0
STL             0
BLK             0
TOV             0
PF              0
PTS_college     0
SOS             0
dtype: int64
```

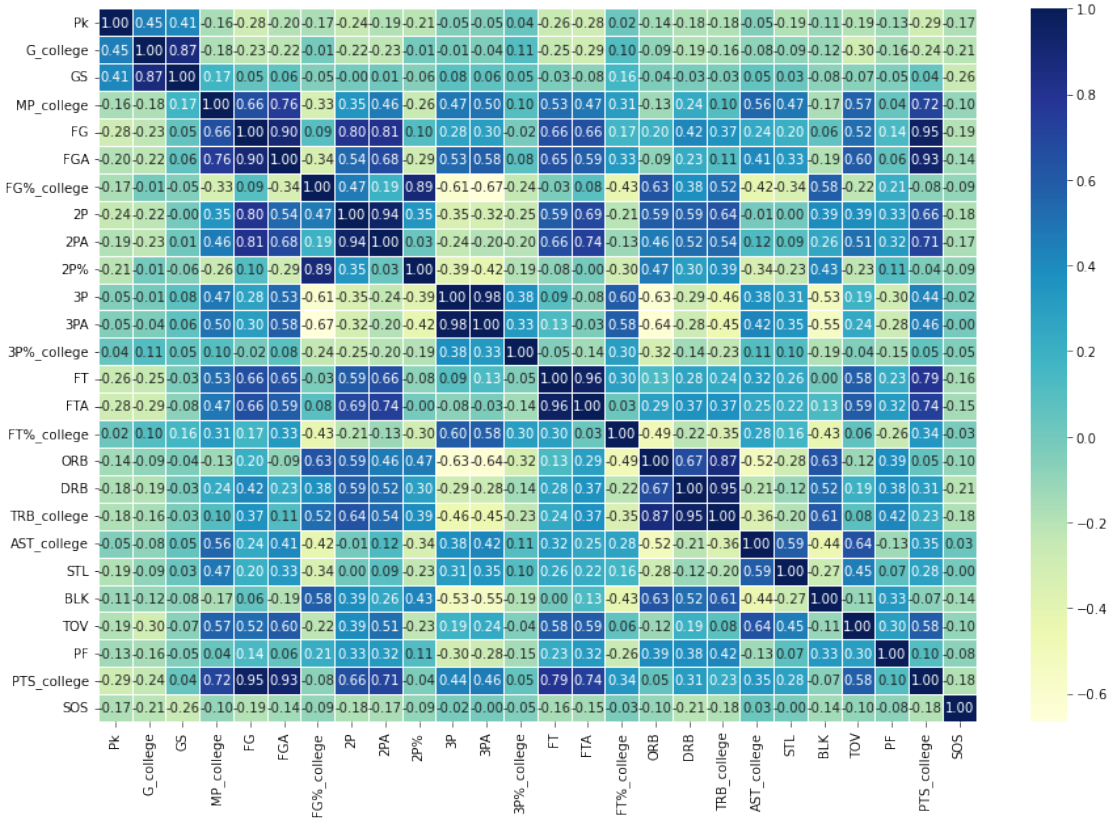
```
draft_data.dropna(inplace=True)
```

```
draft_data.describe()
```

```
                Pk   G_college          GS  MP_college          FG
FGA  \
count  393.000000  393.000000  393.000000  393.000000  393.000000
393.000000
mean    28.974555   81.875318   64.643766   28.463104    4.595674
9.706361
std     17.194694   39.879311   33.876431    4.422506    1.190355
2.610318
min      1.000000    3.000000    0.000000   13.200000    1.300000
2.400000
25%     14.000000   38.000000   35.000000   25.900000    3.700000
7.900000
50%     28.000000   75.000000   65.000000   29.100000    4.600000
9.700000
75%     44.000000  117.000000   89.000000   31.800000    5.300000
11.400000
max     60.000000  152.000000  147.000000   36.500000    9.000000
19.300000

       FG%_college         2P         2PA         2P%  ...         ORB
\
count   393.000000  393.00000  393.000000  393.000000  ...  393.000000

mean      0.478646    3.50229    6.721374    0.520654  ...    1.493893

std       0.059706    1.21442    2.164156    0.058541  ...    0.871290

min       0.356000    0.60000    1.200000    0.374000  ...    0.100000

25%       0.438000    2.60000    5.100000    0.483000  ...    0.800000

50%       0.465000    3.30000    6.500000    0.513000  ...    1.300000

75%       0.514000    4.30000    8.200000    0.553000  ...    2.100000

max       0.769000    8.20000   14.300000    0.800000  ...    4.300000


              DRB  TRB_college  AST_college         STL         BLK  \
count  393.000000   393.000000   393.000000  393.000000  393.000000
mean     3.961323     5.450891     2.246056    1.023410    0.774300
std      1.382057     2.072563     1.510116    0.446485    0.702057
min      1.500000     1.900000     0.200000    0.200000    0.000000
25%      2.900000     3.800000     1.200000    0.700000    0.300000
50%      3.700000     5.100000     1.800000    0.900000    0.500000
75%      4.800000     6.800000     2.900000    1.300000    1.100000
max      8.600000    11.800000     8.700000    2.900000    5.400000
```

```
              TOV          PF  PTS_college         SOS
count  393.000000  393.000000   393.000000  393.000000
mean     1.940967    2.173028    13.086768    7.220967
std      0.659140    0.454103     3.495679    3.190615
min      0.600000    1.000000     3.400000   -5.410000
25%      1.500000    1.900000    10.400000    6.650000
50%      1.900000    2.200000    13.100000    7.840000
75%      2.300000    2.500000    15.500000    9.160000
max      5.200000    4.100000    27.400000   12.750000

[8 rows x 26 columns]
```

```
corr_matrix = draft_data.corr()
plt.figure(figsize=(15, 10))
sns.heatmap(corr_matrix,
            annot=True,
            linewidths=0.5,
            fmt= ".2f",
            cmap="YlGnBu");
```




```
draft_data.drop(["School"],axis=1,inplace=True)

#model with all variables
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

```
from sklearn.model_selection import cross_val_score
np.random.seed(2899)
X = draft_data.drop("Pk",axis=1)
y = draft_data["Pk"]


X_train, X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)

rf=RandomForestRegressor().fit(X_train,y_train)
rfa = rf.score(X_test,y_test)
rfa
```

```
0.2949091184120498
```

```
#ridge regession
from sklearn.linear_model import Ridge
np.random.seed(2899)
ridge=Ridge().fit(X_train,y_train)
ra = ridge.score(X_test,y_test)
ra
```

```
0.2831271077839256
```

```
#lasso regression
from sklearn.linear_model import Lasso
lasso=Lasso().fit(X_train,y_train)
la = lasso.score(X_test,y_test)
la
```

```
0.2883184186274941
```

```
y_preds = rf.predict(X_test)

from sklearn.metrics import mean_squared_error
#rf RMSE
rmse = mean_squared_error(y_test, y_preds, squared=False)
rmse
```

```
13.767979545288906
```

```
y_preds = ridge.predict(X_test)
rmse = mean_squared_error(y_test, y_preds, squared=False)
rmse
```

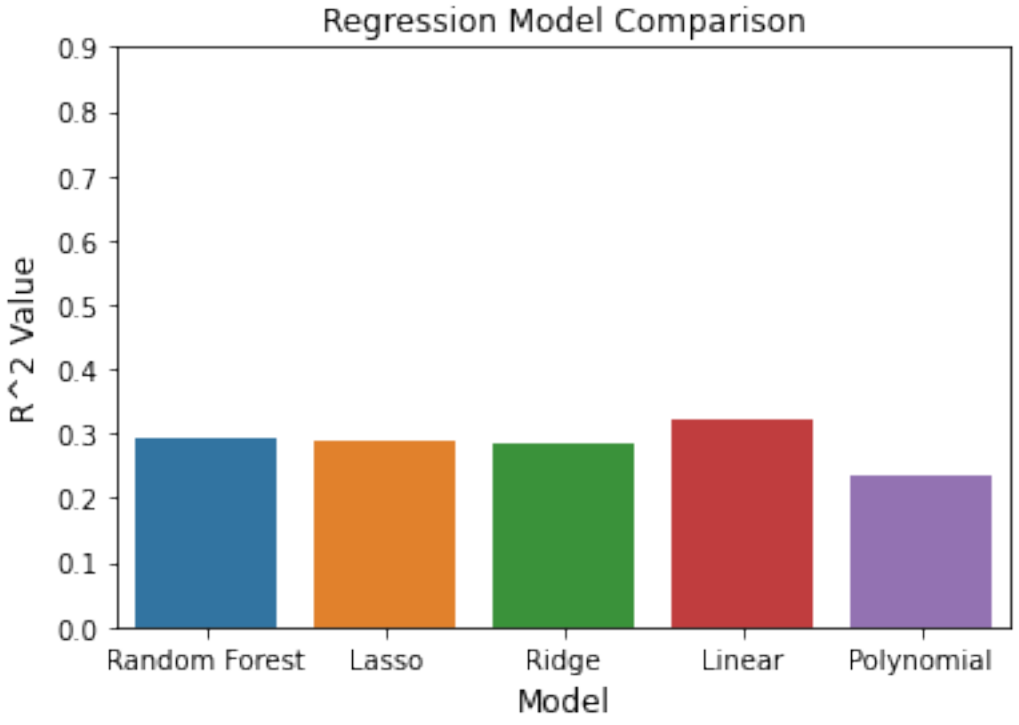
```
13.88253388229793
```

```
y_preds = lasso.predict(X_test)
rmse = mean_squared_error(y_test, y_preds, squared=False)
rmse
```

```
13.832176631359042
```

```
#plt.style.use('fivethirtyeight')
models=["Random Forest","Lasso","Ridge","Linear","Polynomial"]
r2 = [rfa,la,ra,.323,.237]
fig=sns.barplot(models, r2)
fig.set_xlabel("Model", fontsize = 12)
fig.set_ylabel("R^2 Value", fontsize = 12)
fig.set_title("Regression Model Comparison")
fig.set_yticks(np.arange(0, 1, .1))
plt.show()
```

```
/Users/jackpiccione/opt/anaconda3/lib/python3.7/site-packages/
seaborn/_decorators.py:43: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional
argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```




```
#read in data to train lottery model
draft_dataLOT = pd.read_csv("NBAdraft.csv")
```

```
draft_dataLOT.columns
```

```
Index(['Pk', 'School', 'G_college', 'GS', 'MP_college', 'FG', 'FGA',
       'FG%_college', '2P', '2PA', '2P%', '3P', '3PA', '3P%_college',
'FT',
       'FTA', 'FT%_college', 'ORB', 'DRB', 'TRB_college',
'AST_college', 'STL',
```

```
       'BLK', 'TOV', 'PF', 'PTS_college', 'SOS', 'name', 'lottery'],
      dtype='object')
```

```
draft_dataLOT.drop(["Pk","name"],inplace=True,axis=1)
```

```
draft_dataLOT.drop("School",inplace=True,axis=1)
```

```
draft_dataLOT.dropna(inplace=True)
```

```
draft_dataLOT.columns
```

```
Index(['G_college', 'GS', 'MP_college', 'FG', 'FGA', 'FG%_college',
'2P',
       '2PA', '2P%', '3P', '3PA', '3P%_college', 'FT', 'FTA', 'FT
%_college',
       'ORB', 'DRB', 'TRB_college', 'AST_college', 'STL', 'BLK',
'TOV', 'PF',
       'PTS_college', 'SOS', 'lottery'],
      dtype='object')
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import Ridge
np.random.seed(2899)
X = draft_dataLOT.drop("lottery",axis=1)
y = draft_dataLOT["lottery"]

X_train, X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)

rfc=RandomForestClassifier().fit(X_train,y_train)

rfc.score(X_test, y_test)
```

```
0.7341772151898734
```

```
from sklearn.ensemble import GradientBoostingClassifier
np.random.seed(2899)
X = draft_dataLOT.drop("lottery",axis=1)
y = draft_dataLOT["lottery"]

X_train, X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)

gbc=GradientBoostingClassifier().fit(X_train,y_train)

gbc.score(X_test, y_test)
```

```
0.759493670886076
```

```
prospects = pd.read_csv("prospects.csv")

prospects.head()
```

```
    G  GS    MP   FG   FGA    FG%   2P  2PA    2P%   3P  ...  DRB  TRB
AST  \
```

```
0  32  31  26.9  5.3   8.8  0.607  4.0  5.5  0.737  1.3  ...  8.1  9.9
1.9
1  39  39  33.0  6.3  13.2  0.478  5.2  9.8  0.525  1.1  ...  6.1  7.8
3.2
2  29  29  25.3  4.9   8.1  0.597  4.9  8.1  0.600  0.0  ...  5.2  8.1
1.3
3  34  34  32.0  3.2   8.4  0.384  1.4  3.4  0.426  1.8  ...  3.2  4.0
1.4
4  39  25  24.0  3.7   7.6  0.493  1.9  3.5  0.547  1.8  ...  3.2  3.9
1.0

   STL  BLK  TOV   PF   PTS    SOS            name
0  0.8  3.7  1.9  2.7  14.1   4.50   chet-holmgren
1  1.1  0.9  2.4  1.9  17.2   7.26  paolo-banchero
2  0.8  2.1  2.2  2.7  12.0   7.86     jalen-duren
3  0.7  0.2  1.5  1.7  10.1  11.02   caleb-houstan
4  0.5  0.6  0.6  1.1  10.4   7.26      aj-griffin

[5 rows x 26 columns]
```

```
chl=prospects.loc[[0]].values.flatten().tolist()
name=chl.pop()
achl=[chl]
ynew = rfc.predict(achl)
ynew,name
```

```
(array([ True]), 'chet-holmgren')
```

```
prospects.dropna(inplace=True)
```

```
prospects=prospects.reset_index(drop=True)
```

```
df_list5=[]
df=[]
for i in range(len(prospects)):
    plist=prospects.loc[[i]].values.flatten().tolist()
    name=plist.pop()
    parray=[plist]
    ynew = gbc.predict_proba(parray)
    ylist=ynew.tolist()
    ylist=ylist[0]
    #df["name"] = plist.pop() #add player name to dataframe
    #df["lottery_prob"] = ylist.pop()
    df.append(name)
    df.append(ylist.pop())
    #df_list5.append(df)
    print(ynew,name)
```

```
[[0.13694653 0.86305347]] chet-holmgren
[[0.35865844 0.64134156]] paolo-banchero
[[0.32427004 0.67572996]] jalen-duren
```

[[0.96324324 0.03675676]] caleb-houstan
[[0.48338117 0.51661883]] aj-griffin
[[0.8507817 0.1492183]] patrick-baldwinjr
[[0.21480413 0.78519587]] jaden-ivey
[[0.85142094 0.14857906]] peyton-watson
[[0.76896919 0.23103081]] tyty-washingtonjr
[[0.49508863 0.50491137]] kennedy-chandler
[[0.14717415 0.85282585]] nolan-hickman
[[0.60389648 0.39610352]] jd-davison
[[0.98268289 0.01731711]] allen-flanigan
[[0.97854359 0.02145641]] johnny-juzang
[[0.75616143 0.24383857]] marcus-bagley
[[0.79591399 0.20408601]] julian-champagnie
[[0.70837284 0.29162716]] gabe-brown
[[0.89705985 0.10294015]] mark-williams
[[0.62267199 0.37732801]] matthew-mayer
[[0.66721309 0.33278691]] jabari-walker
[[0.16258231 0.83741769]] keegan-murray
[[0.7946896 0.2053104]] max-abmas
[[0.73710987 0.26289013]] caleb-love
[[0.81059227 0.18940773]] ochai-agbaji
[[0.96737749 0.03262251]] collin-gillespie
[[0.64409835 0.35590165]] kendall-brown
[[0.94542676 0.05457324]] matthew-cleveland
[[0.8549522 0.1450478]] walker-kessler
[[0.98181191 0.01818809]] taevion-kinsey
[[0.96630583 0.03369417]] jamaree-bouyea
[[0.61291264 0.38708736]] drew-timme
[[0.96847587 0.03152413]] josiah-jordan-james
[[0.63746849 0.36253151]] trayce-jackson-davis
[[0.96069979 0.03930021]] dalen-terry
[[0.46760666 0.53239334]] malaki-branham
[[0.92952245 0.07047755]] jalen-wilson
[[0.87886422 0.12113578]] ron-harperjr
[[0.13742008 0.86257992]] bryce-mcgowens
[[0.93222982 0.06777018]] ej-liddell
[[0.91489753 0.08510247]] scotty-pippenjr
[[0.29446372 0.70553628]] kofi-cockburn
[[0.97613121 0.02386879]] taran-armstrong
[[0.99069255 0.00930745]] isaiah-mobley
[[0.87483201 0.12516799]] jahvon-quinerly
[[0.99200043 0.00799957]] marcus-sasser
[[0.80705167 0.19294833]] david-roddy
[[0.83822783 0.16177217]] dereon-seabron
[[0.87271505 0.12728495]] hyunjung-lee
[[0.81327779 0.18672221]] justin-lewis
[[0.77390809 0.22609191]] terquavion-smith
[[0.85138221 0.14861779]] jaylin-williams
[[0.86513647 0.13486353]] jaylin-williams
[[0.94872858 0.05127142]] tevin-brown

```
[[0.98827423 0.01172577]] iverson-molinar
[[0.0846073 0.9153927]] keon-ellis
[[0.9859432 0.0140568]] jordan-hall
[[0.94123388 0.05876612]] christian-braun
[[0.9266844 0.0733156]] ryan-rollins
[[0.83856882 0.16143118]] blake-wesley
[[0.9858812 0.0141188]] jake-laravia
[[0.9477659 0.0522341]] pete-nance
[[0.98205569 0.01794431]] trevion-williams
[[0.85314669 0.14685331]] josh-minott
[[0.98792285 0.01207715]] andrew-nembhard
[[0.5372853 0.4627147]] tari-eason
[[0.73939688 0.26060312]] trevor-keels
[[0.75953322 0.24046678]] julian-strawther
[[0.98186583 0.01813417]] wendell-moorejr
[[0.98686309 0.01313691]] alondes-williams
[[0.82137082 0.17862918]] christian-koloko
[[0.34967901 0.65032099]] jeremy-sochan
[[0.92336123 0.07663877]] harrison-ingram
[[0.97574703 0.02425297]] tyler-burton
[[0.36153445 0.63846555]] jabari-smith
[[0.81151773 0.18848227]] johnny-davis
```

```
def every_second_element(values):
    second_values = []

    for index in range(1, len(values), 2):
        second_values.append(values[index])

    return second_values

lot_prob_list=every_second_element(df)

res = [i for i in df if i not in lot_prob_list]

len(res)
```

```
75
```

```
len(lot_prob_list)
```

```
75
```

```
d = {'Prospect':res,'Lottery_Probability':lot_prob_list}

lottery = pd.DataFrame(d)

lottery.sort_values(by=['Lottery_Probability'],ascending=False).head(1
5)
```

```
            Prospect  Lottery_Probability
54        keon-ellis             0.915393
```

```
0      chet-holmgren             0.863053
37    bryce-mcgowens             0.862580
10     nolan-hickman             0.852826
20     keegan-murray             0.837418
6         jaden-ivey             0.785196
40     kofi-cockburn             0.705536
2        jalen-duren             0.675730
70     jeremy-sochan             0.650321
1     paolo-banchero             0.641342
73      jabari-smith             0.638466
34    malaki-branham             0.532393
4         aj-griffin             0.516619
9   kennedy-chandler             0.504911
64        tari-eason             0.462715
```

```
feat_importances = pd.Series(gbc.feature_importances_,
index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.style.use('fivethirtyeight')
plt.title("Top 10 important features")
plt.show()
```