Data Description

This dataset features the salaries of 612 NHL players for the 2016/2017 season.
There are 153 predictor columns as well as a leading column with the players 2016/2017 annual salary.

Column Legend

(Acronym - Meaning)

Categorical features

Born - Birth date City - City of birth
Cntry - Country of birth Pr/St - Province or state of birth
Nat - Nationality
Hand - Handedness First Name - Last Name - Team -
Position - Positions played. NHL source listed first, followed by those listed by any other source.

Numeric Feature

%FOT - Percentage of all on-ice faceoffs taken by this player.
+/- - Plus/minus
1G - First goals of a game
A/60 - Events Against per 60 minutes, defaults to Corsi, but can be set to another stat
A1 - First assists, primary assists
A2 - Second assists, secondary assists
BLK% - Percentage of all opposing shot attempts blocked by this player
C.Close - A player shot attempt (Corsi) differential when the game was close
C.Down - A player shot attempt (Corsi) differential when the team was trailing
C.Tied - A player shot attempt (Corsi) differential when the team was tied
C.Up - A player shot attempt (Corsi) differential when the team was in the lead
CA - Shot attempts allowed (Corsi, SAT) while this player was on the ice
Cap Hit - The player's cap hit
CBar - Crossbars hit
CF - The team's shot attempts (Corsi, SAT) while this player was on the ice
CF.QoC - A weighted average of the Corsi percentage of a player's opponents
CF.QoT - A weighted average of the Corsi percentage of a player's linemates
CHIP - Cap Hit of Injured Player is games lost to injury multiplied by cap hit per game
DAP - Disciplined aggression proxy, which is hits and takeaways divided by minor penalties
DFA - Dangerous Fenwick against, which is on-ice unblocked shot attempts weighted by shot quality
DFF - Dangerous Fenwick for, which is on-ice unblocked shot attempts weighted by shot quality
DFF.QoC - Quality of Competition metric based on Dangerous Fenwick, which is unblocked shot attempts weighted for shot quality DftRd - Round in which the player was drafted

DftYr - Year drafted
Diff - Events for minus event against, defaults to Corsi, but can be set to another stat
Diff/60 - Events for minus event against, per 60 minutes, defaults to Corsi, but can be set to another stat
DPS - Defensive point shares, a catch-all stats that measures a player's defensive contributions in points in the standings DSA - Dangerous shots allowed while this player was on the ice, which is rebounds plus rush shots
DSF - The team's dangerous shots while this player was on the ice, which is rebounds plus rush shots
DZF - Shifts this player has ended with an defensive zone faceoff
dzFOL - Faceoffs lost in the defensive zone
dzFOW - Faceoffs win in the defensive zone
dzGAPF - Team goals allowed after faceoffs taken in the defensive zone
dzGFPF - Team goals scored after faceoffs taken in the defensive zone
DZS - Shifts this player has started with an defensive zone faceoff
dzSAPF - Team shot attempts allowed after faceoffs taken in the defensive zone
dzSFPF - Team shot attempts taken after faceoffs taken in the defensive zone * 345678E+/- - A player's expected +/-, based on his team and minutes played
ENG - Empty-net goals
Exp dzNGPF - Expected goal differential after faceoffs taken in the defensive zone, based on the number of them
Exp dzNSPF - Expected shot differential after faceoffs taken in the defensive zone, based on the number of them
Exp ozNGPF - Expected goal differential after faceoffs taken in the offensive zone, based on the number of them
Exp ozNSPF - Expected shot differential after faceoffs taken in the offensive zone, based on the number of them
F.Close - A player unblocked shot attempt (Fenwick) differential when the game was close
F.Down - A player unblocked shot attempt (Fenwick) differential when the team was trailing
F.Tied - A player unblocked shot attempt (Fenwick) differential when the team was tied
F.Up - A player unblocked shot attempt (Fenwick) differential when the team was in the lead. Not the best acronym.
F/60 - Events For per 60 minutes, defaults to Corsi, but can be set to another stat
FA - Unblocked shot attempts allowed (Fenwick, USAT) while this player was on the ice
FF - The team's unblocked shot attempts (Fenwick, USAT) while this player was on the ice
First Name -
FO% - Faceoff winning percentage
FO%vsL - Faceoff winning percentage against lefthanded opponents
FO%vsR - Faceoff winning percentage against righthanded opponents
FOL - The team's faceoff losses while this player was on the ice
FOL.Close - Faceoffs lost when the score was close
FOL.Down - Faceoffs lost when the team was trailing
FOL.Up - Faceoffs lost when the team was in the lead
FovsL - Faceoffs taken against lefthanded opponents
FovsR - Faceoffs taken against righthanded opponents
FOW - The team's faceoff wins while this player was on the ice

FOW.Close - Faceoffs won when the score was close
FOW.Down - Faceoffs won when the team was trailing
FOW.Up - Faceoffs won when the team was in the lead
G - Goals
G.Bkhd - Goals scored on the backhand
G.Dflct - Goals scored with deflections
G.Slap - Goals scored with slap shots
G.Snap - Goals scored with snap shots
G.Tip - Goals scored with tip shots
G.Wrap - Goals scored with a wraparound
G.Wrst - Goals scored with a wrist shot
GA - Goals allowed while this player was on the ice
Game - Game Misconduct penalties
GF - The team's goals while this player was on the ice
GP - Games Played
Grit - Defined as hits, blocked shots, penalty minutes, and majors
GS - The player's combined game score
GS/G - The player's average game score
GVA - The team's giveaways while this player was on the ice
GWG - Game-winning goals
GWG - Game-winning goals
HA - The team's hits taken while this player was on the ice
HF - The team's hits thrown while this player was on the ice
HopFO - Opening faceoffs taken at home
HopFOW - Opening faceoffs won at home
Ht - Height
iBLK - Shots blocked by this individual
iCF - Shot attempts (Corsi, SAT) taken by this individual
iDS - Dangerous shots taken by this player, the sum of rebounds and shots off the rush
iFF - Unblocked shot attempts (Fenwick, USAT) taken by this individual
iFOL - Faceoff losses by this individual
iFOW - Faceoff wins by this individual
iGVA - Giveaways by this individual
iHA - Hits taken by this individual
iHDf - The difference in hits thrown by this individual minus those taken
iHF - Hits thrown by this individual
iMiss - Individual shots taken that missed the net.
Injuries - List of types of injuries incurred, if any
iPEND - Penalties drawn by this individual
iPenDf - The difference in penalties drawn minus those taken
iPENT - Penalties taken by this individual
IPP% - Individual points percentage, which is on-ice goals for which this player had the goal or an assist
iRB - Rebound shots taken by this individual
iRS - Shots off the rush taken by this individual
iSCF - All scoring chances taken by this individual

iSF - Shots on goal taken by this individual
iTKA - Takeaways by this individual
ixG - Expected goals (weighted shots) for this individual, which is shot attempts weighted by shot location
Maj - Major penalties taken
Match - Match penalties
MGL - Games lost due to injury
Min - Minor penalties taken
Misc - Misconduct penalties
NGPF - Net Goals Post Faceoff. A differential of all goals within 10 seconds of a faceoff, relative to expectations set by the zone in which they took place
NHLid - NHL player id useful when looking at the raw data in game files
NMC - What kind of no-movement clause this player's contract has, if any
NPD - Net Penalty Differential is the player's penalty differential relative to a player of the same position with the same ice time per manpower situation
NSPF - Net Shots Post Faceoff. A differential of all shot attempts within 10 seconds of a faceoff, relative to expectations set by the zone in which they took place
NZF - Shifts this player has ended with a neutral zone faceoff
nzFOL - Faceoffs lost in the neutral zone
nzFOW - Faceoffs won in the neutral zone
nzGAPF - Team goals allowed after faceoffs taken in the neutral zone
nzGFPF - Team goals scored after faceoffs taken in the neutral zone
NZS - Shifts this player has started with a neutral zone faceoff
nzSAPF - Team shot attempts allowed after faceoffs taken in the neutral zone
nzSFPF - Team shot attempts taken after faceoffs taken in the neutral zone
OCA - Shot attempts allowed (Corsi, SAT) while this player was not on the ice
OCF - The team's shot attempts (Corsi, SAT) while this player was not on the ice
ODZS - Defensive zone faceoffs that occurred without this player on the ice
OFA - Unblocked shot attempts allowed (Fenwick, USAT) while this player was not on the ice
OFF - The team's unblocked shot attempts (Fenwick, USAT) while this player was not on the ice
OGA - Goals allowed while this player was not on the ice
OGF - The team's goals while this player was not on the ice
ONZS - Neutral zone faceoffs that occurred without this player on the ice
OOZS - Offensive zone faceoffs that occurred without this player on the ice
OpFO - Opening faceoffs taken
OpFOW - Opening faceoffs won
OppCA60 - A weighted average of the shot attempts (Corsi, SAT) the team allowed per 60 minutes of a player's opponents
OppCF60 - A weighted average of the shot attempts (Corsi, SAT) the team generated per 60 minutes of a player's opponents
OppFA60 - A weighted average of the unblocked shot attempts (Fenwick, USAT) the team allowed per 60 minutes of a player's opponents
OppFF60 - A weighted average of the unblocked shot attempts (Fenwick, USAT) the team generated per 60 minutes of a player's opponents
OppGA60 - A weighted average of the goals the team allowed per 60 minutes of a player's

opponents
OppGF60 - A weighted average of the goals the team scored per 60 minutes of a player's opponents
OppSA60 - A weighted average of the shots on goal the team allowed per 60 minutes of a player's opponents
OppSF60 - A weighted average of the shots on goal the team generated per 60 minutes of a player's opponents
OPS - Offensive point shares, a catch-all stats that measures a player's offensive contributions in points in the standings
OSA - Shots on goal allowed while this player was not on the ice
OSCA - Scoring chances allowed while this player was not on the ice
OSCF - The team's scoring chances while this player was not on the ice
OSF - The team's shots on goal while this player was not on the ice
OTF - Shifts this player started with an on-the-fly change
OTG - Overtime goals
OTOI - The amount of time this player was not on the ice.
Over - Shots that went over the net
Ovrl - Where the player was drafted overall
OxGA - Expected goals allowed (weighted shots) while this player was not on the ice, which is shot attempts weighted by location OxGF - The team's expected goals (weighted shots) while this player was not on the ice, which is shot attempts weighted by location
OZF - Shifts this player has ended with an offensive zone faceoff
ozFO - Faceoffs taken in the offensive zone
ozFOL - Faceoffs lost in the offensive zone
ozFOW - Faceoffs won in the offensive zone
ozGAPF - Team goals allowed after faceoffs taken in the offensive zone
ozGFPF - Team goals scored after faceoffs taken in the offensive zone
OZS - Shifts this player has started with an offensive zone faceoff
ozSAPF - Team shot attempts allowed after faceoffs taken in the offensive zone
ozSFPF - Team shot attempts taken after faceoffs taken in the offensive zone
Pace - The average game pace, as estimated by all shot attempts per 60 minutes
Pass - An estimate of the player's setup passes (passes that result in a shot attempt)
Pct% - Percentage of all events produced by this team, defaults to Corsi, but can be set to another stat
PDO - The team's shooting and save percentages added together, times a thousand
PEND - The team's penalties drawn while this player was on the ice
PENT - The team's penalties taken while this player was on the ice
PIM - Penalties in minutes
Post - Times hit the post
PS - Point shares, a catch-all stats that measures a player's contributions in points in the standings
PSA - Penalty shot attempts
PSG - Penalty shot goals
PTS - Points. Goals plus all assists
PTS/60 - Points per 60 minutes
QRelCA60 - Shot attempts allowed per 60 minutes relative to how others did against the

same competition
QRelCF60 - Shot attempts per 60 minutes relative to how others did against the same competition
QRelDFA60 - Weighted unblocked shot attempts (Dangeorus Fenwick) allowed per 60 minutes relative to how others did against the same competition
QRelDFF60 - Weighted unblocked shot attempts (Dangeorus Fenwick) per 60 minutes relative to how others did against the same competition
RBA - Rebounds allowed while this player was on the ice. Two very different sources.
RBF - The team's rebounds while this player was on the ice. Two very different sources.
RelA/60 - The player's A/60 relative to the team when he's not on the ice
RelC/60 - Corsi differential per 60 minutes relative to his team
RelC% - Corsi percentage relative to his team
RelDf/60 - The player's Diff/60 relative to the team when he's not on the ice
RelF/60 - The player's F/60 relative to the team when he's not on the ice
RelF/60 - Fenwick differential per 60 minutes relative to his team
RelF% - Fenwick percentage relative to his team
RelPct% - The players Pct% relative to the team when he's not on the ice
RelZS% - The player's zone start percentage when he's on the ice relative to when he's not.
RopFO - Opening faceoffs taken at home
RopFOW - Opening faceoffs won at home
RSA - Shots off the rush allowed while this player was on the ice
RSF - The team's shots off the rush while this player was on the ice
S.Bkhd - Backhand shots
S.Dflct - Deflections
S.Slap - Slap shots
S.Snap - Snap shots
S.Tip - Tipped shots
S.Wrap - Wraparound shots
S.Wrst - Wrist shots
SA - Shots on goal allowed while this player was on the ice
Salary - The player's salary
SCA - Scoring chances allowed while this player was on the ice
SCF - The team's scoring chances while this player was on the ice
sDist - The average shot distance of shots taken by this player
SF - The team's shots on goal while this player was on the ice
SH% - The team's (not individual's) shooting percentage when the player was on the ice
SOG - Shootout Goals
SOGDG - Game-deciding shootout goals
SOS - Shootout Shots
Status - This player's free agency status
SV% - The team's save percentage when the player was on the ice
TKA - The team's takeaways while this player was on the ice
TMCA60 - A weighted average of the shot attempts (Corsi, SAT) the team allowed per 60 minutes of a player's linemates
TMCF60 - A weighted average of the shot attempts (Corsi, SAT) the team generated per 60 minutes of a player's linemates

TMFA60 - A weighted average of the unblocked shot attempts (Fenwick, USAT) the team allowed per 60 minutes of a player's linemates
TMFF60 - A weighted average of the unblocked shot attempts (Fenwick, USAT) the team generated per 60 minutes of a player's linemates
TMGA60 - A weighted average of the goals the team allowed per 60 minutes of a player's linemates
TMGF60 - A weighted average of the goals the team scored per 60 minutes of a player's linemates
TMSA60 - A weighted average of the shots on goal the team allowed per 60 minutes of a player's linemates
TMSF60 - A weighted average of the shots on goal the team generated per 60 minutes of a player's linemates
TmxGF - A weighted average of a player's linemates of the expected goals the team scored
TmxGA - A weighted average of a player's linemates of the expected goals the team allowed
TMGA - A weighted average of a player's linemates of the goals the team scored
TMGF - A weighted average of a player's linemates of the goals the team allowed
TOI - Time on ice, in minutes, or in seconds (NHL)
TOI.QoC - A weighted average of the TOI% of a player's opponents.
TOI.QoT - A weighted average of the TOI% of a player's linemates.
TOI/GP - Time on ice divided by games played
TOI% - Percentage of all available ice time assigned to this player.
Wide - Shots that went wide of the net
Wt - Weight
xGA - Expected goals allowed (weighted shots) while this player was on the ice, which is shot attempts weighted by location
xGF - The team's expected goals (weighted shots) while this player was on the ice, which is shot attempts weighted by location xGF.QoC - A weighted average of the expected goal percentage of a player's opponents
xGF.QoT - A weighted average of the expected goal percentage of a player's linemates
ZS% - Zone start percentage, the percentage of shifts started in the offensive zone, not counting neutral zone or on-the-fly changes

Aim

To predict NHL player's salaries for the test dataset. What are the best predictors of how much a player will make?

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data=pd.read_csv('C:/Users/harshita/Downloads/NHL Players/train.csv', encoding='IS(
```

In [3]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [4]:
```
data.head()
```

Out[4]:

	Salary	Born	City	Pr/St	Cntry	Nat	Ht	Wt	DftYr	DftRd	Ovrl	Hand	Last Name
0	925000	97-01-30	Sainte-Marie	QC	CAN	CAN	74	190	2015.0	1.0	18.0	L	Chabot
1	2250000	93-12-21	Ottawa	ON	CAN	CAN	74	207	2012.0	1.0	15.0	R	Ceci
2	8000000	88-04-16	St. Paul	MN	USA	USA	72	218	2006.0	1.0	7.0	R	Okposo
3	3500000	92-01-07	Ottawa	ON	CAN	CAN	77	220	2010.0	1.0	3.0	R	Gudbranson
4	1750000	94-03-29	Toronto	ON	CAN	CAN	76	217	2012.0	1.0	16.0	R	Wilson

In [5]:
```
data.shape
```

Out[5]:
```
(612, 154)
```

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 612 entries, 0 to 611
Columns: 154 entries, Salary to GS/G
dtypes: float64(73), int64(71), object(10)
memory usage: 736.4+ KB
```

In [7]:
```
data.info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 612 entries, 0 to 611
Data columns (total 154 columns):
 #    Column      Non-Null Count  Dtype
---   ------      --------------  -----
 0    Salary      612 non-null    int64
 1    Born        612 non-null    object
 2    City        612 non-null    object
 3    Pr/St       459 non-null    object
 4    Cntry       612 non-null    object
 5    Nat         612 non-null    object
 6    Ht          612 non-null    int64
 7    Wt          612 non-null    int64
 8    DftYr       512 non-null    float64
 9    DftRd       512 non-null    float64
 10   Ovrl        512 non-null    float64
 11   Hand        612 non-null    object
 12   Last Name   612 non-null    object
 13   First Name  612 non-null    object
 14   Position    612 non-null    object
 15   Team        612 non-null    object
 16   GP          612 non-null    int64
 17   G           612 non-null    int64
 18   A           612 non-null    int64
 19   A1          612 non-null    int64
 20   A2          612 non-null    int64
 21   PTS         612 non-null    int64
 22   +/-         612 non-null    int64
 23   E+/-        612 non-null    float64
 24   PIM         612 non-null    int64
 25   Shifts      612 non-null    int64
 26   TOI         612 non-null    int64
 27   TOIX        611 non-null    float64
 28   TOI/GP      612 non-null    float64
 29   TOI/GP.1    612 non-null    float64
 30   TOI%        611 non-null    float64
 31   IPP%        611 non-null    float64
 32   SH%         610 non-null    float64
 33   SV%         611 non-null    float64
 34   PDO         610 non-null    float64
 35   F/60        611 non-null    float64
 36   A/60        611 non-null    float64
 37   Pct%        612 non-null    float64
 38   Diff        612 non-null    int64
 39   Diff/60     611 non-null    float64
 40   iCF         604 non-null    float64
 41   iCF.1       612 non-null    int64
 42   iFF         604 non-null    float64
 43   iSF         604 non-null    float64
 44   iSF.1       612 non-null    int64
 45   iSF.2       612 non-null    int64
 46   ixG         611 non-null    float64
 47   iSCF        611 non-null    float64
 48   iRB         604 non-null    float64
 49   iRS         611 non-null    float64
 50   iDS         605 non-null    float64
 51   sDist       612 non-null    float64
 52   sDist.1     596 non-null    float64
 53   Pass        611 non-null    float64
 54   iHF         612 non-null    int64
 55   iHF.1       611 non-null    float64
 56   iHA         603 non-null    float64
 57   iHDf        604 non-null    float64
 58   iMiss       612 non-null    int64
```

59	iGVA	612	non-null	int64
60	iTKA	612	non-null	int64
61	iBLK	612	non-null	int64
62	iGVA.1	611	non-null	float64
63	iTKA.1	611	non-null	float64
64	iBLK.1	611	non-null	float64
65	BLK%	611	non-null	float64
66	iFOW	612	non-null	int64
67	iFOL	612	non-null	int64
68	iFOW.1	611	non-null	float64
69	iFOL.1	611	non-null	float64
70	FO%	612	non-null	float64
71	%FOT	611	non-null	float64
72	dzFOW	612	non-null	int64
73	dzFOL	612	non-null	int64
74	nzFOW	612	non-null	int64
75	nzFOL	612	non-null	int64
76	ozFOW	612	non-null	int64
77	ozFOL	612	non-null	int64
78	FOW.Up	612	non-null	int64
79	FOL.Up	612	non-null	int64
80	FOW.Down	612	non-null	int64
81	FOL.Down	612	non-null	int64
82	FOW.Close	612	non-null	int64
83	FOL.Close	612	non-null	int64
84	OTG	612	non-null	int64
85	1G	612	non-null	int64
86	GWG	612	non-null	int64
87	ENG	612	non-null	int64
88	PSG	612	non-null	int64
89	PSA	612	non-null	int64
90	G.Bkhd	612	non-null	int64
91	G.Dflct	612	non-null	int64
92	G.Slap	612	non-null	int64
93	G.Snap	612	non-null	int64
94	G.Tip	612	non-null	int64
95	G.Wrap	612	non-null	int64
96	G.Wrst	612	non-null	int64
97	CBar	612	non-null	int64
98	Post	612	non-null	int64
99	Over	612	non-null	int64
100	Wide	612	non-null	int64
101	S.Bkhd	612	non-null	int64
102	S.Dflct	612	non-null	int64
103	S.Slap	612	non-null	int64
104	S.Snap	612	non-null	int64
105	S.Tip	612	non-null	int64
106	S.Wrap	612	non-null	int64
107	S.Wrst	612	non-null	int64
108	iPenT	612	non-null	int64
109	iPenD	612	non-null	int64
110	iPENT	611	non-null	float64
111	iPEND	611	non-null	float64
112	iPenDf	612	non-null	int64
113	NPD	612	non-null	float64
114	Min	612	non-null	int64
115	Maj	612	non-null	int64
116	Match	612	non-null	int64
117	Misc	612	non-null	int64
118	Game	612	non-null	int64
119	CF	611	non-null	float64
120	CA	611	non-null	float64
121	FF	611	non-null	float64
122	FA	611	non-null	float64

```
 123  SF          611 non-null    float64
 124  SA          611 non-null    float64
 125  xGF         611 non-null    float64
 126  xGA         611 non-null    float64
 127  SCF         611 non-null    float64
 128  SCA         611 non-null    float64
 129  GF          611 non-null    float64
 130  GA          611 non-null    float64
 131  RBF         611 non-null    float64
 132  RBA         611 non-null    float64
 133  RSF         611 non-null    float64
 134  RSA         611 non-null    float64
 135  DSF         612 non-null    int64
 136  DSA         612 non-null    int64
 137  FOW         611 non-null    float64
 138  FOL         611 non-null    float64
 139  HF          611 non-null    float64
 140  HA          611 non-null    float64
 141  GVA         611 non-null    float64
 142  TKA         611 non-null    float64
 143  PENT        611 non-null    float64
 144  PEND        611 non-null    float64
 145  OPS         612 non-null    float64
 146  DPS         612 non-null    float64
 147  PS          612 non-null    float64
 148  OTOI        611 non-null    float64
 149  Grit        612 non-null    int64
 150  DAP         612 non-null    float64
 151  Pace        611 non-null    float64
 152  GS          611 non-null    float64
 153  GS/G        610 non-null    float64
dtypes: float64(73), int64(71), object(10)
memory usage: 736.4+ KB
```

In [8]:
```
data.describe(include='all')
```

Out[8]:

	Salary	Born	City	Pr/St	Cntry	Nat	Ht	Wt	DftYr	
count	6.120000e+02	612	612	459	612	612	612.00000	612.000000	512.000000	51
unique	NaN	576	373	37	18	16	NaN	NaN	NaN	
top	NaN	93-03-26	Toronto	ON	CAN	CAN	NaN	NaN	NaN	
freq	NaN	3	26	136	291	289	NaN	NaN	NaN	
mean	2.264509e+06	NaN	NaN	NaN	NaN	NaN	72.98366	200.745098	2008.787109	
std	2.236340e+06	NaN	NaN	NaN	NaN	NaN	2.08016	14.952420	4.440032	
min	5.750000e+05	NaN	NaN	NaN	NaN	NaN	67.00000	160.000000	1990.000000	
25%	7.425000e+05	NaN	NaN	NaN	NaN	NaN	72.00000	190.000000	2006.000000	
50%	9.250000e+05	NaN	NaN	NaN	NaN	NaN	73.00000	200.000000	2010.000000	
75%	3.500000e+06	NaN	NaN	NaN	NaN	NaN	74.00000	210.000000	2012.000000	
max	1.380000e+07	NaN	NaN	NaN	NaN	NaN	81.00000	265.000000	2016.000000	

Data Cleaning

In [9]:
```
data.isnull().sum()
```

Out[9]:

Salary	0
Born	0
City	0
Pr/St	153
Cntry	0
Nat	0
Ht	0
Wt	0
DftYr	100
DftRd	100
Ovrl	100
Hand	0
Last Name	0
First Name	0
Position	0
Team	0
GP	0
G	0
A	0
A1	0
A2	0
PTS	0
+/-	0
E+/-	0
PIM	0
Shifts	0
TOI	0
TOIX	1
TOI/GP	0
TOI/GP.1	0
TOI%	1
IPP%	1
SH%	2
SV%	1
PDO	2
F/60	1
A/60	1
Pct%	0
Diff	0
Diff/60	1
iCF	8
iCF.1	0
iFF	8
iSF	8
iSF.1	0
iSF.2	0
ixG	1
iSCF	1
iRB	8
iRS	1
iDS	7
sDist	0
sDist.1	16
Pass	1
iHF	0
iHF.1	1
iHA	9
iHDf	8
iMiss	0
iGVA	0
iTKA	0
iBLK	0
iGVA.1	1
iTKA.1	1

```
iBLK.1          1
BLK%            1
iFOW            0
iFOL            0
iFOW.1          1
iFOL.1          1
FO%             0
%FOT            1
dzFOW           0
dzFOL           0
nzFOW           0
nzFOL           0
ozFOW           0
ozFOL           0
FOW.Up          0
FOL.Up          0
FOW.Down        0
FOL.Down        0
FOW.Close       0
FOL.Close       0
OTG             0
1G              0
GWG             0
ENG             0
PSG             0
PSA             0
G.Bkhd          0
G.Dflct         0
G.Slap          0
G.Snap          0
G.Tip           0
G.Wrap          0
G.Wrst          0
CBar            0
Post            0
Over            0
Wide            0
S.Bkhd          0
S.Dflct         0
S.Slap          0
S.Snap          0
S.Tip           0
S.Wrap          0
S.Wrst          0
iPenT           0
iPenD           0
iPENT           1
iPEND           1
iPenDf          0
NPD             0
Min             0
Maj             0
Match           0
Misc            0
Game            0
CF              1
CA              1
FF              1
FA              1
SF              1
SA              1
xGF             1
xGA             1
SCF             1
```

```
SCA             1
GF              1
GA              1
RBF             1
RBA             1
RSF             1
RSA             1
DSF             0
DSA             0
FOW             1
FOL             1
HF              1
HA              1
GVA             1
TKA             1
PENT            1
PEND            1
OPS             0
DPS             0
PS              0
OTOI            1
Grit            0
DAP             0
Pace            1
GS              1
GS/G            2
dtype: int64
```

From the above data info we can see that the dataset with numerical features contains number of null values and one categorical feature contains null values.

creating a list of features with null values 1 or 2

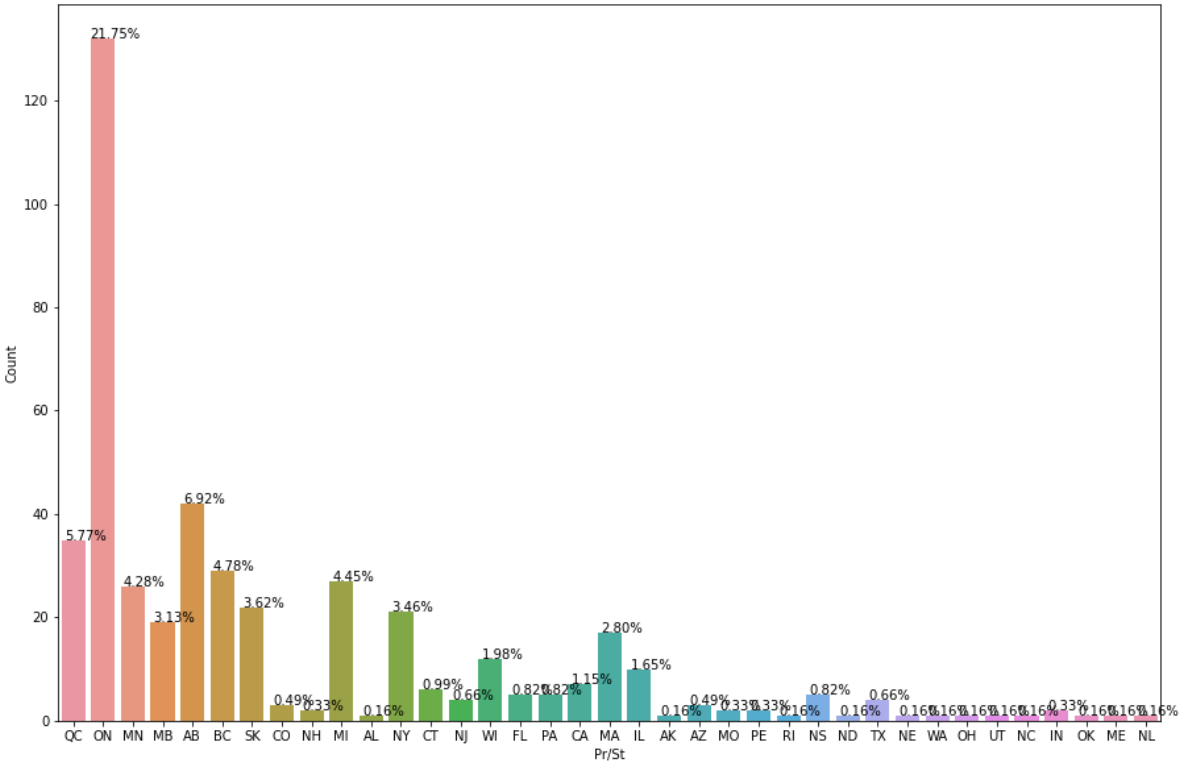
In [11]:
```
num_nan=['TOIX','TOI%','IPP%','SH%','SV%','PDO','F/60','A/60','Diff/60','ixG','iSCI
```

In [12]:
```
print(len(num_nan))
data1=data.copy()
print(data1.shape)
data.dropna(subset=num_nan,inplace=True)
data.shape
```

Out[12]:
```
51
(612, 154)
(607, 154)
```

Checking for the data distribution under Pr/St to replace its null values

In [13]:
```
plt.figure(figsize=(15,10))
ax=sns.countplot(x="Pr/St",data=data)
plt.xlabel("Pr/St")
plt.ylabel("Count")
for p in ax.patches:
    percentage='{:.2f}%'.format(100*p.get_height()/607)
    x=p.get_x()+p.get_width()
    y=p.get_height()
    ax.annotate(percentage,(x,y),ha='center')
plt.show()
```



From the above graph it could be seen that ON has the highest distribution in Pr/St

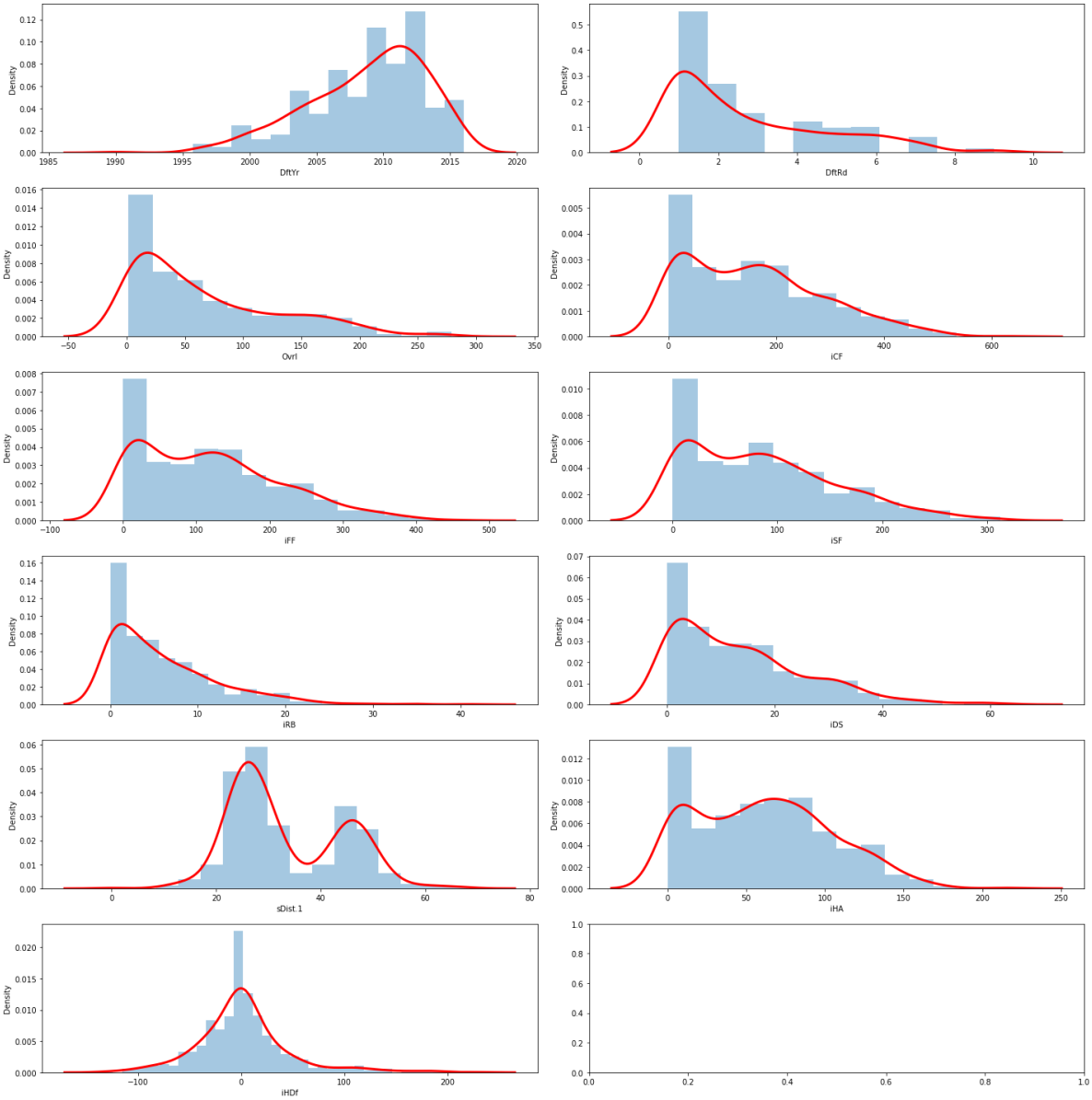
In [14]:
```
data['Pr/St'] = data['Pr/St'].replace(np.nan, 'ON')
```

creating a list of features with null values above 6

In [15]:
```
num_nan_2=['DftYr','DftRd','Ovrl','iCF','iFF','iSF','iRB','iDS','sDist.1','iHA','iH
```

Plotting distribution curse for num_nan_2 features in order to check their skewness

In [16]:
```
rows=6
cols=2
fig,ax=plt.subplots(nrows=rows,ncols=cols,figsize=(20,20));
col=num_nan_2
index=0
for i in range(rows):
    for j in range(cols):
        sns.distplot(data[col[index]],ax=ax[i][j],kde_kws={'linewidth':3,'color':'r
        index+=1
        if index>9:
            break
plt.tight_layout();
plt.show();
```



From the above distibution plots we can see that the only feature to be symmetric in nature is iHDf while all other features are highly skewed

In [17]:
```
data['iHDf'].fillna(data['iHDf'].mean(), inplace=True)
```

In [18]:
```
for i in range(0,10):
    data[num_nan_2[i]].fillna(data[num_nan_2[i]].median(), inplace=True)
```

In [19]:
```
data.isnull().sum().any()
```

Out[19]:
```
False
```

There are no missing values left in our data now

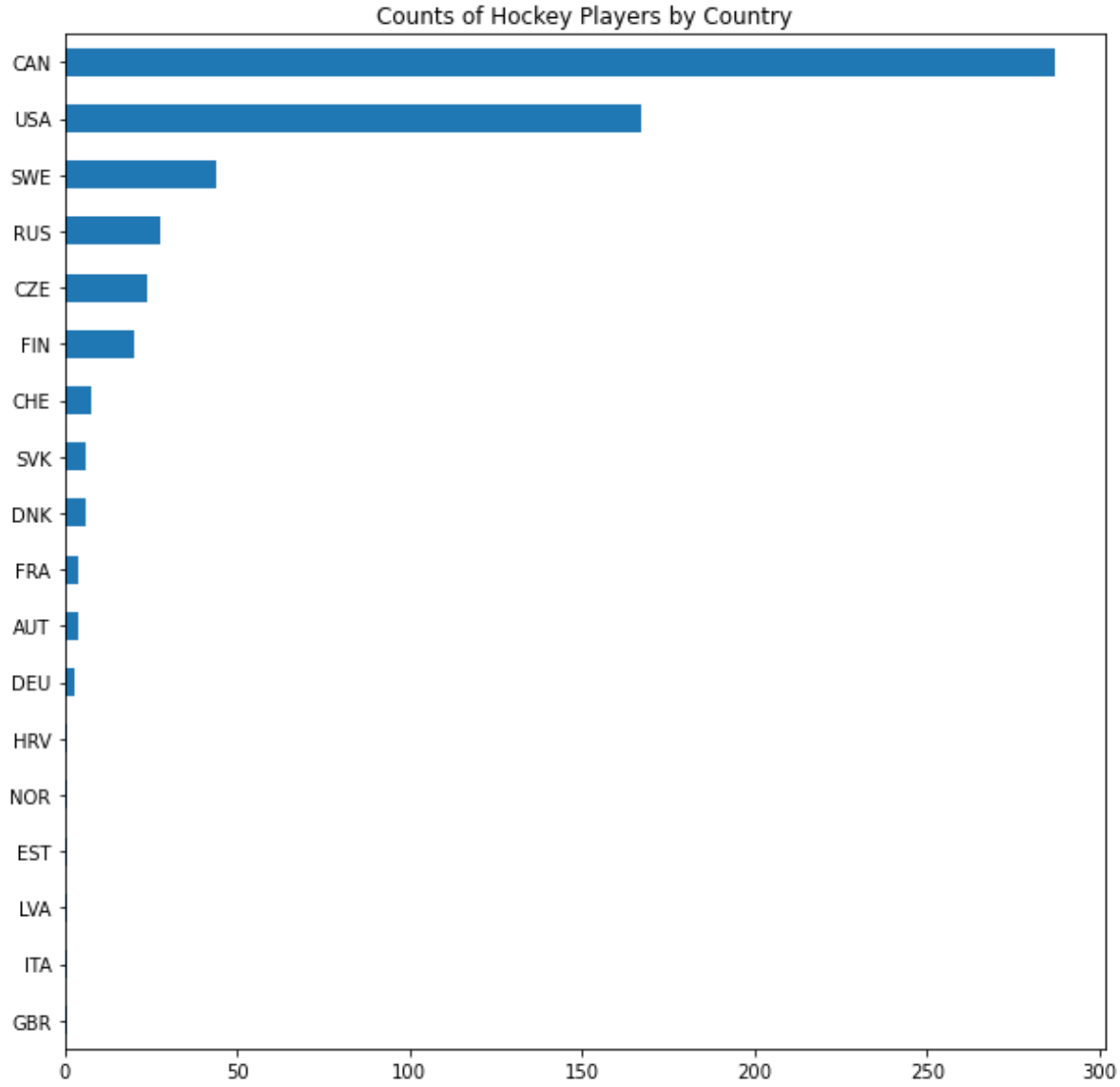
In [20]:
```
data.shape
```

Out[20]:
```
(607, 154)
```

Exploratory Data Analysis

In [21]:
```
fig, ax=plt.subplots(1,1,figsize=(10,10))
data['Cntry'].value_counts().sort_values().plot(kind='barh',ax=ax)
```

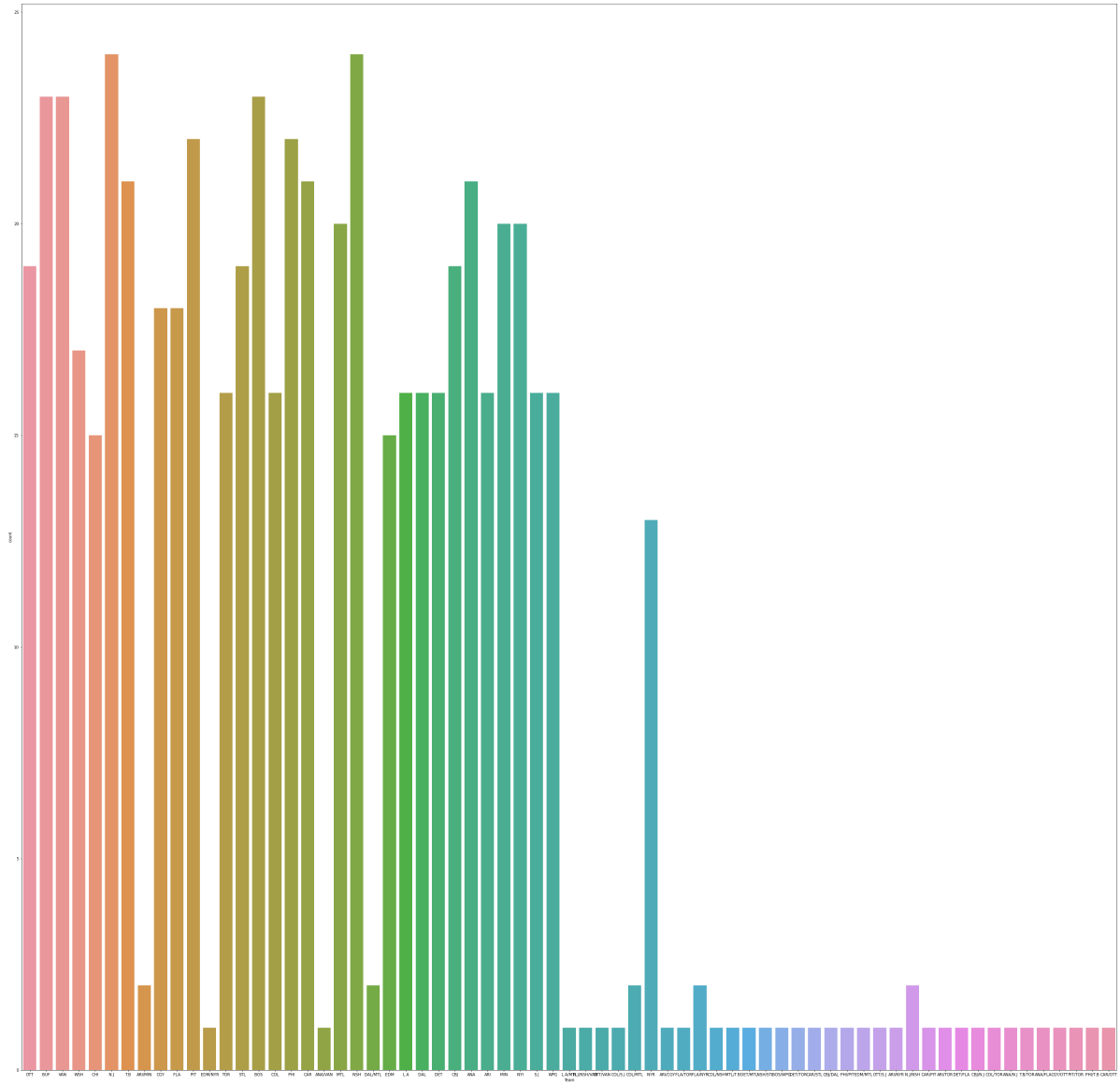
```
ax.set_title("Counts of Hockey Players by Country");
```



Maximum player are from Canada

In [22]:
```
plt.figure(figsize=(50,50))
sns.countplot(x = "Team", data = data)
```

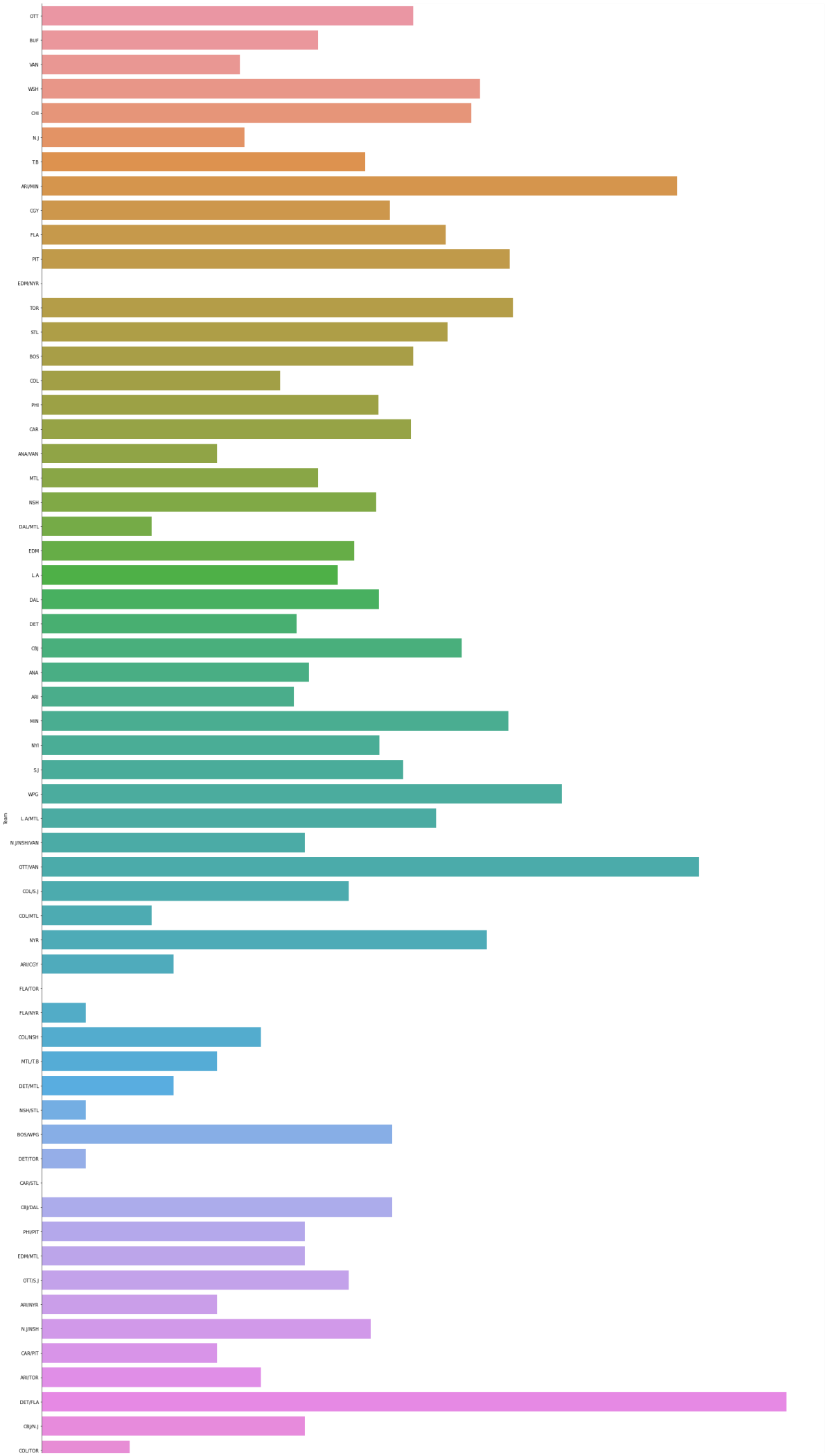
Out[22]:
```
<AxesSubplot:xlabel='Team', ylabel='count'>
```

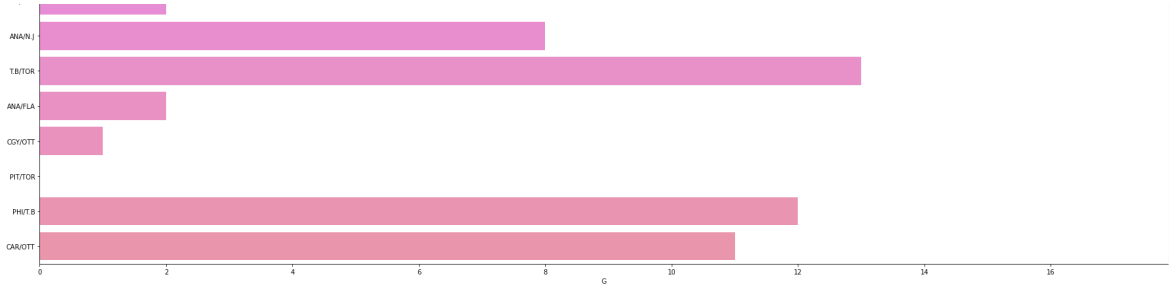


Maximum Player count are from Team NSH

In [23]:
```
sns.catplot(x='G',y='Team', kind = 'bar', data = data , height=50,aspect =0.5,ci=N
```

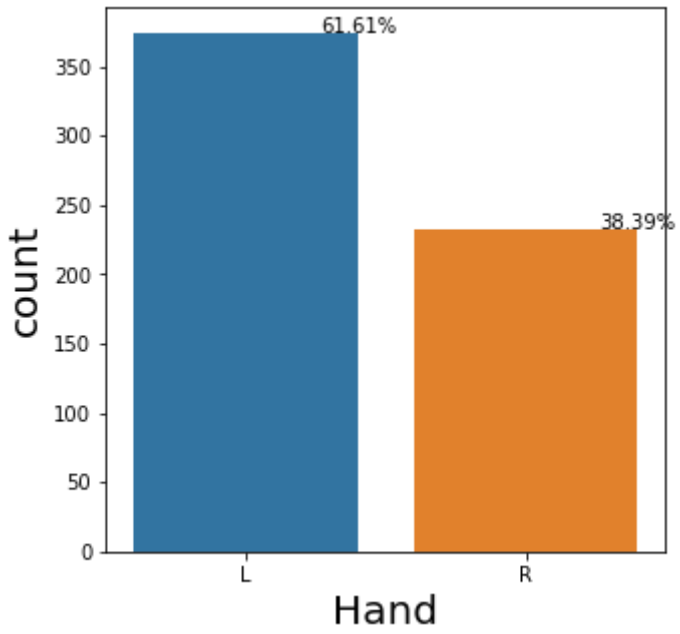
Out[23]:
```
<seaborn.axisgrid.FacetGrid at 0x225fb100190>
```


Team
OTT
BUF
VAN
WSH
CHI
N.J
TB
ARI/MIN
CGY
FLA
PIT
EDM/NYR
TOR
STL
BOS
COL
PHI
CAR
ANA/VAN
MTL
NSH
DAL/MTL
EDM
L.A
DAL
DET
CBJ
ANA
ARI
MIN
NYI
S.J
WPG
L.A/MTL
N.J/NSH/VAN
OTT/VAN
COL/S.J
COL/MTL
NYR
ARI/CGY
FLA/TOR
FLA/NYR
COL/NSH
MTL/T.B
DET/MTL
NSH/STL
BOS/WPG
DET/TOR
CAR/STL
CBJ/DAL
PHI/PIT
EDM/MTL
OTT/S.J
ARI/NYR
N.J/NSH
CAR/PIT
ARI/TOR
DET/FLA
CBJ/N.J
COL/TOR



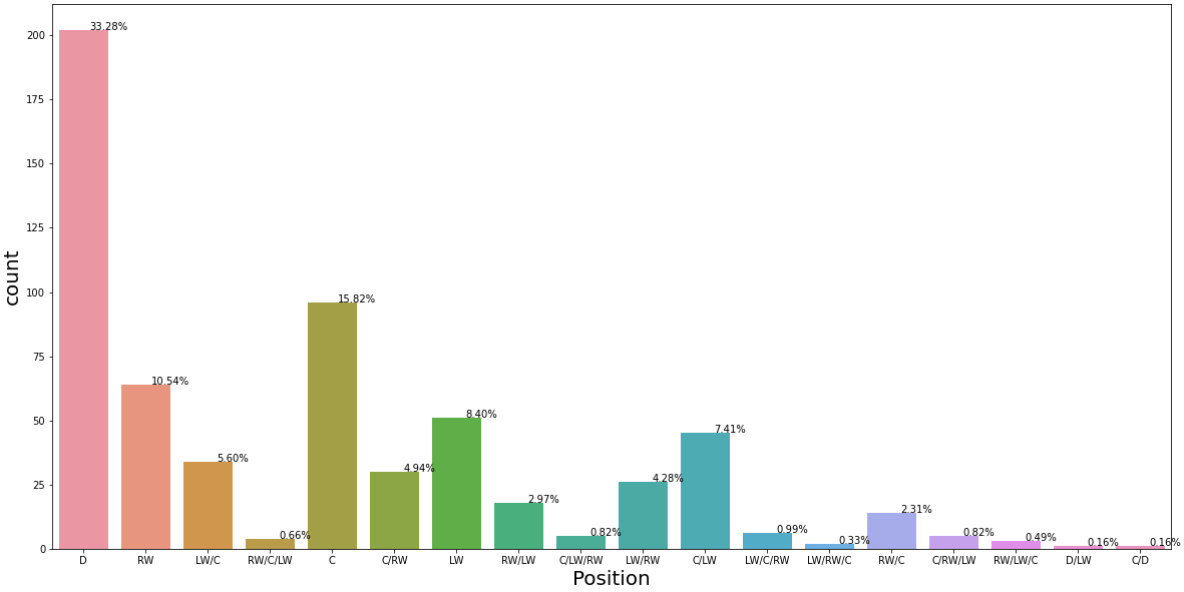
Maximum goals are accounted by the team DET/FLA

In [24]:
```
plt.figure(figsize = (5,5))
ax=sns.countplot(x = "Hand", data = data)
plt.xlabel("Hand",fontsize=20)
plt.ylabel("count", fontsize=20)

for p in ax.patches:
    percentage='{:.2f}%'.format(100*p.get_height()/607)
    x=p.get_x()+p.get_width()
    y=p.get_height()
    ax.annotate(percentage,(x,y),ha='center')
plt.show()
```



From the above count plot we can see that we have more number of left handed players as compare to right handed

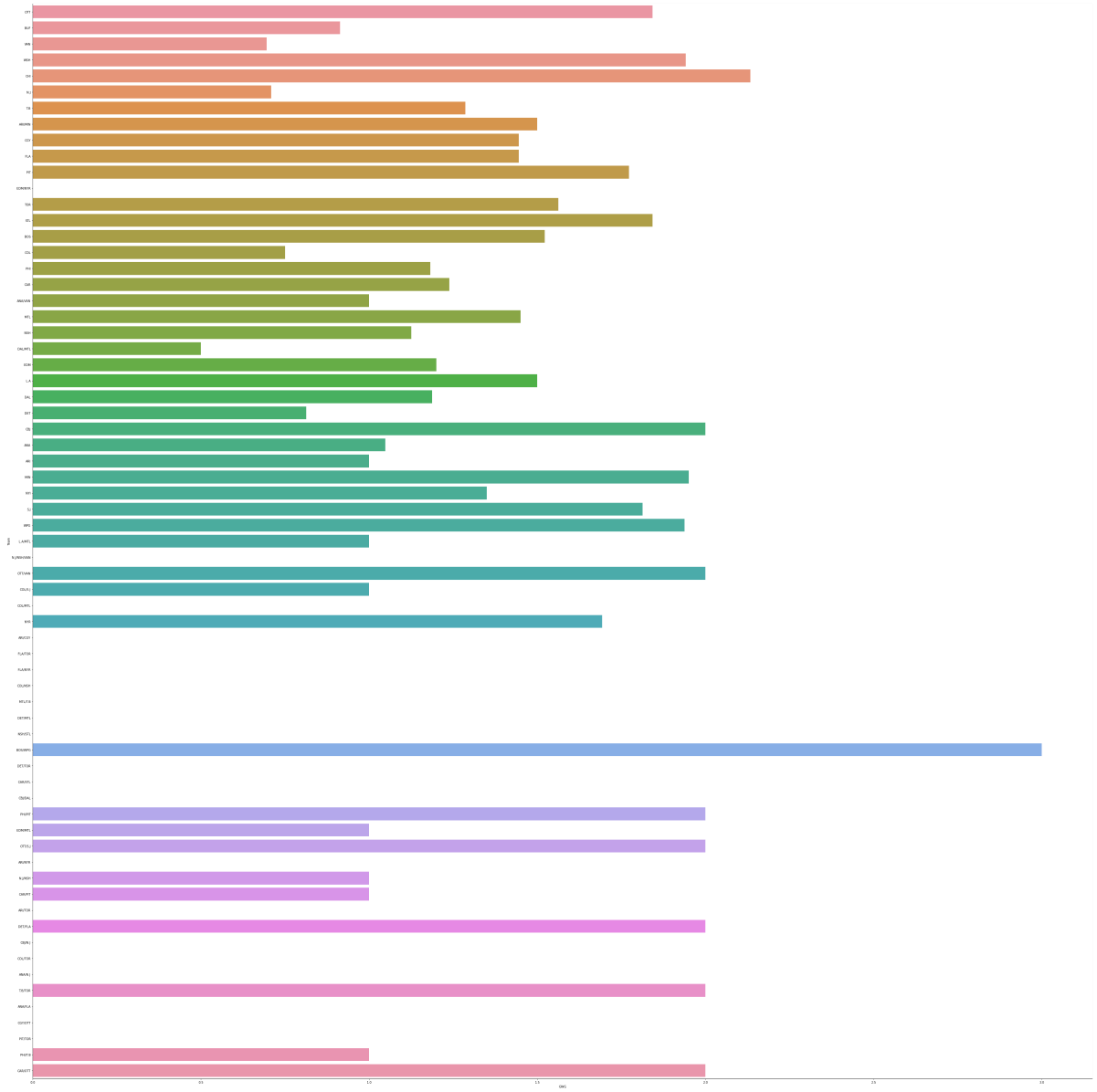
In [25]:
```
plt.figure(figsize = (20,10))
ax=sns.countplot(x = "Position", data = data)
plt.xlabel("Position",fontsize=20)
plt.ylabel("count", fontsize=20)
for p in ax.patches:
    percentage='{:.2f}%'.format(100*p.get_height()/607)
    x=p.get_x()+p.get_width()
    y=p.get_height()
    ax.annotate(percentage,(x,y),ha='center')
plt.show()
```



From the above count plot we can see that we have more number of players in defence position

In [26]:
```
sns.catplot(x='GWG',y='Team', kind = 'bar', data = data , height=50,aspect =1.0,ci
```

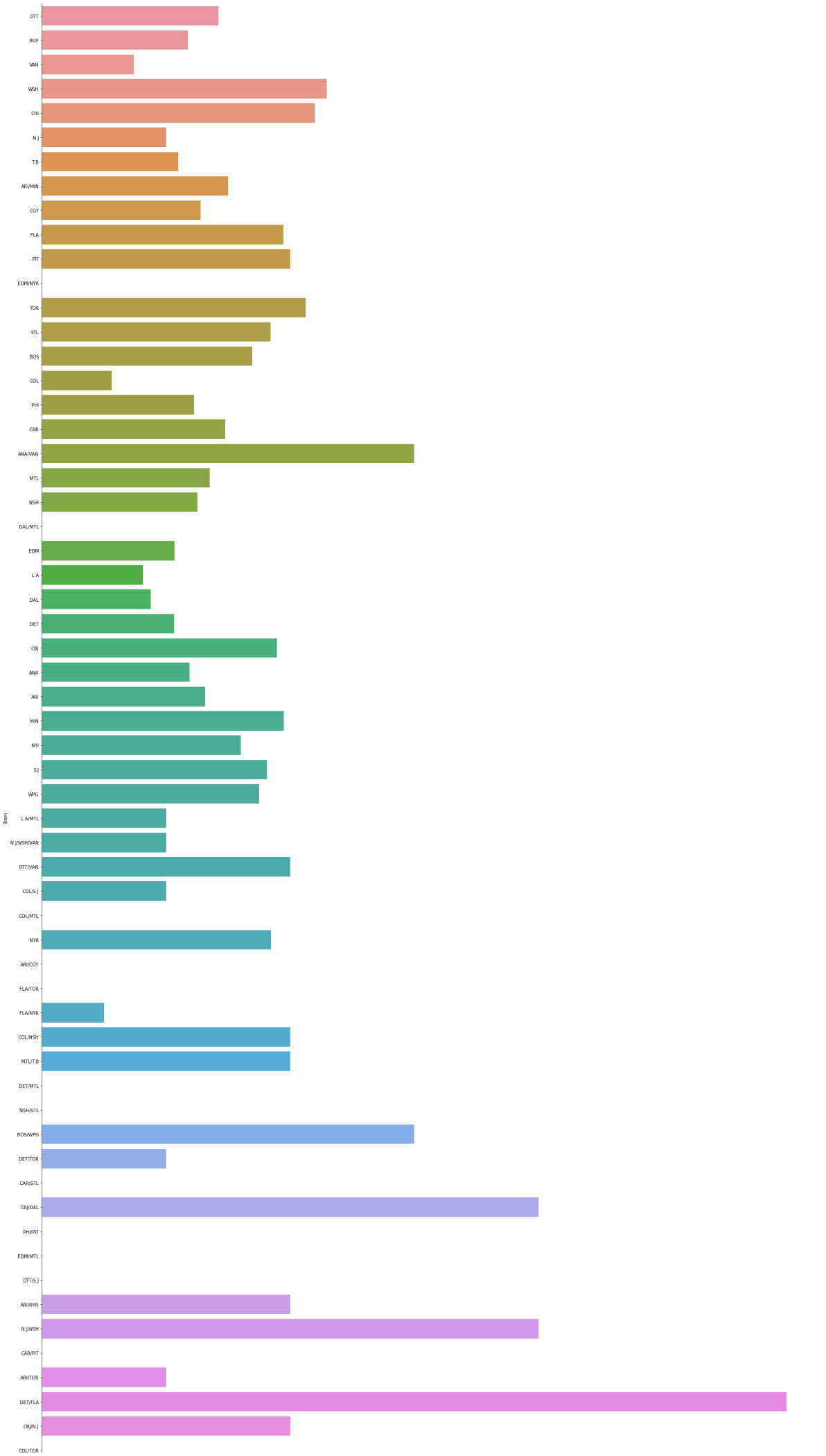
Out[26]:
```
<seaborn.axisgrid.FacetGrid at 0x225ff20fc70>
```

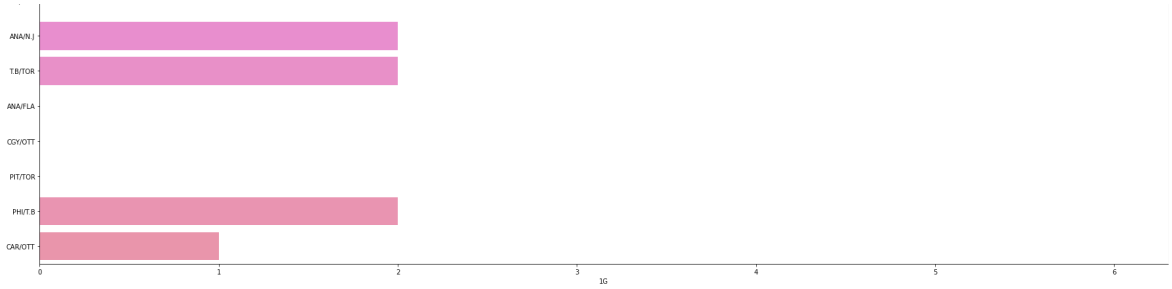


The BOS/WPG team has scored the highest number of game willing goals

In [27]:
```
sns.catplot(x='1G',y='Team', kind = 'bar', data = data , height=50,aspect =0.5,ci=
```

Out[27]:
```
<seaborn.axisgrid.FacetGrid at 0x225fa199eb0>
```

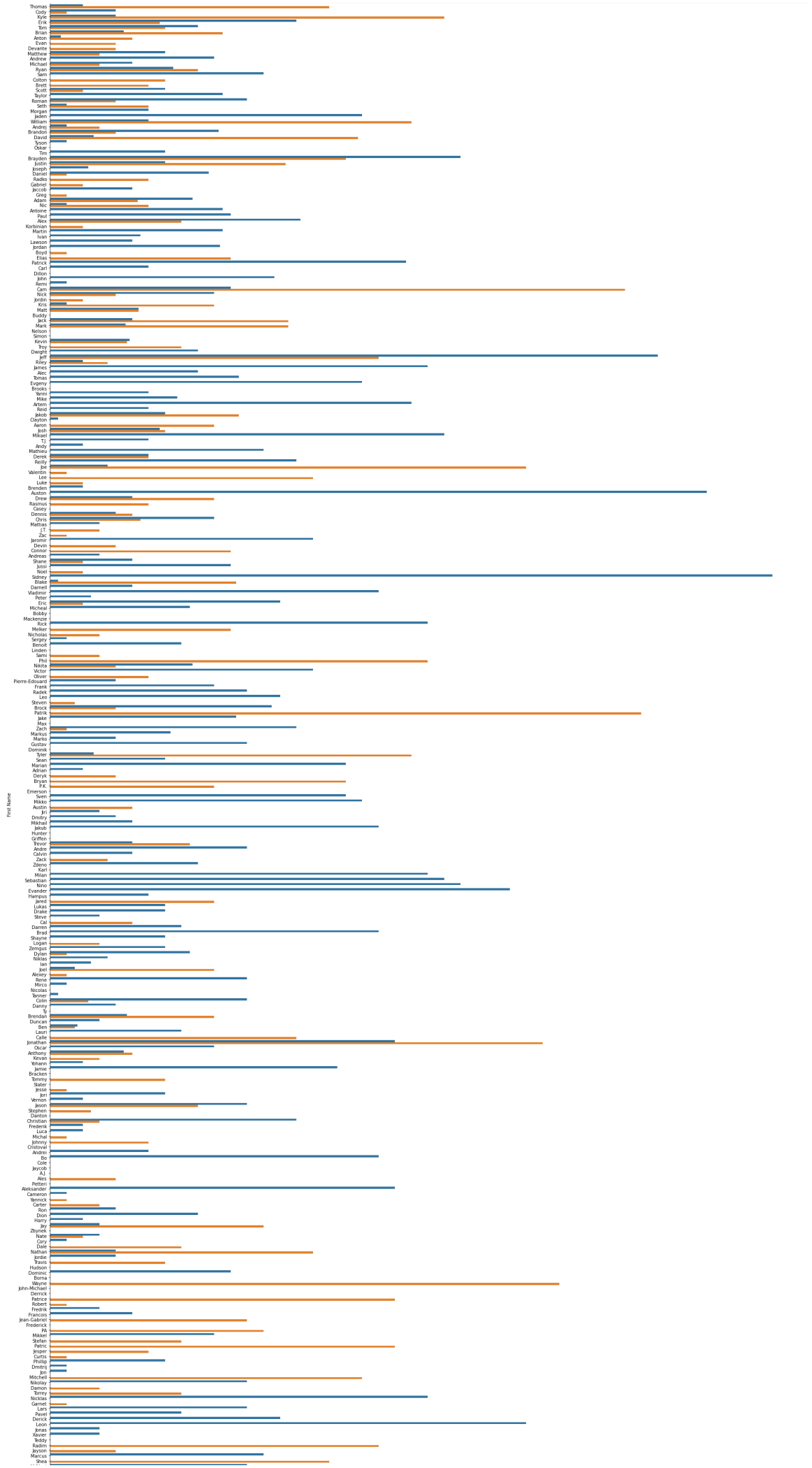

Team
OTT
BUF
VAN
WSH
CHI
N.J
TB
ARI/MIN
CGY
FLA
PIT
EDM/NYR
TOR
STL
BOS
COL
PHI
CAR
ANA/VAN
MTL
NSH
DAL/MTL
EDM
L.A
DAL
DET
CBJ
ANA
ARI
MIN
NYI
S.J
WPG
L.A/MTL
N.J/NSH/VAN
OTT/VAN
COL/S.J
COL/MTL
NYR
ARI/CGY
FLA/TOR
FLA/NYR
COL/NSH
MTL/T.B
DET/MTL
NSH/STL
BOS/WPG
DET/TOR
CAR/STL
CBJ/DAL
PHI/PIT
EDM/MTL
OTT/S.J
ARI/NYR
N.J/NSH
CAR/PIT
ARI/TOR
DET/FLA
CBJ/N.J
COL/TOR

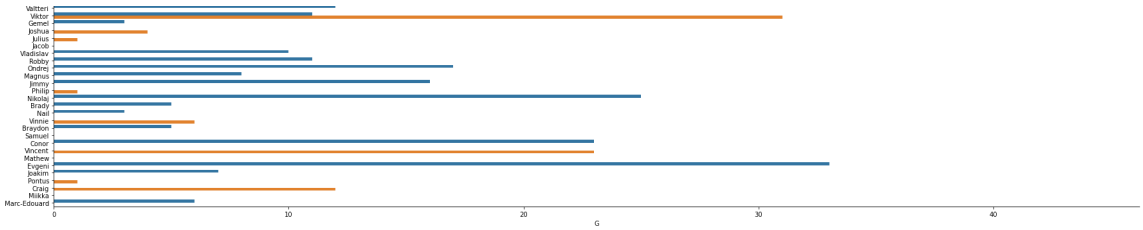


DET/FLA has scored first goal in different matches maximum number of times

In [28]:
```
plt.figure(figsize=(100,100))
sns.catplot(x='G',y='First Name', kind = 'bar',hue= 'Hand', data = data , height=5
#set_xticklabels(labels=list of labels on x-axis,rotation,rotation=True)
```

Out[28]:
```
<seaborn.axisgrid.FacetGrid at 0x225fb3f95e0>

<Figure size 7200x7200 with 0 Axes>
```

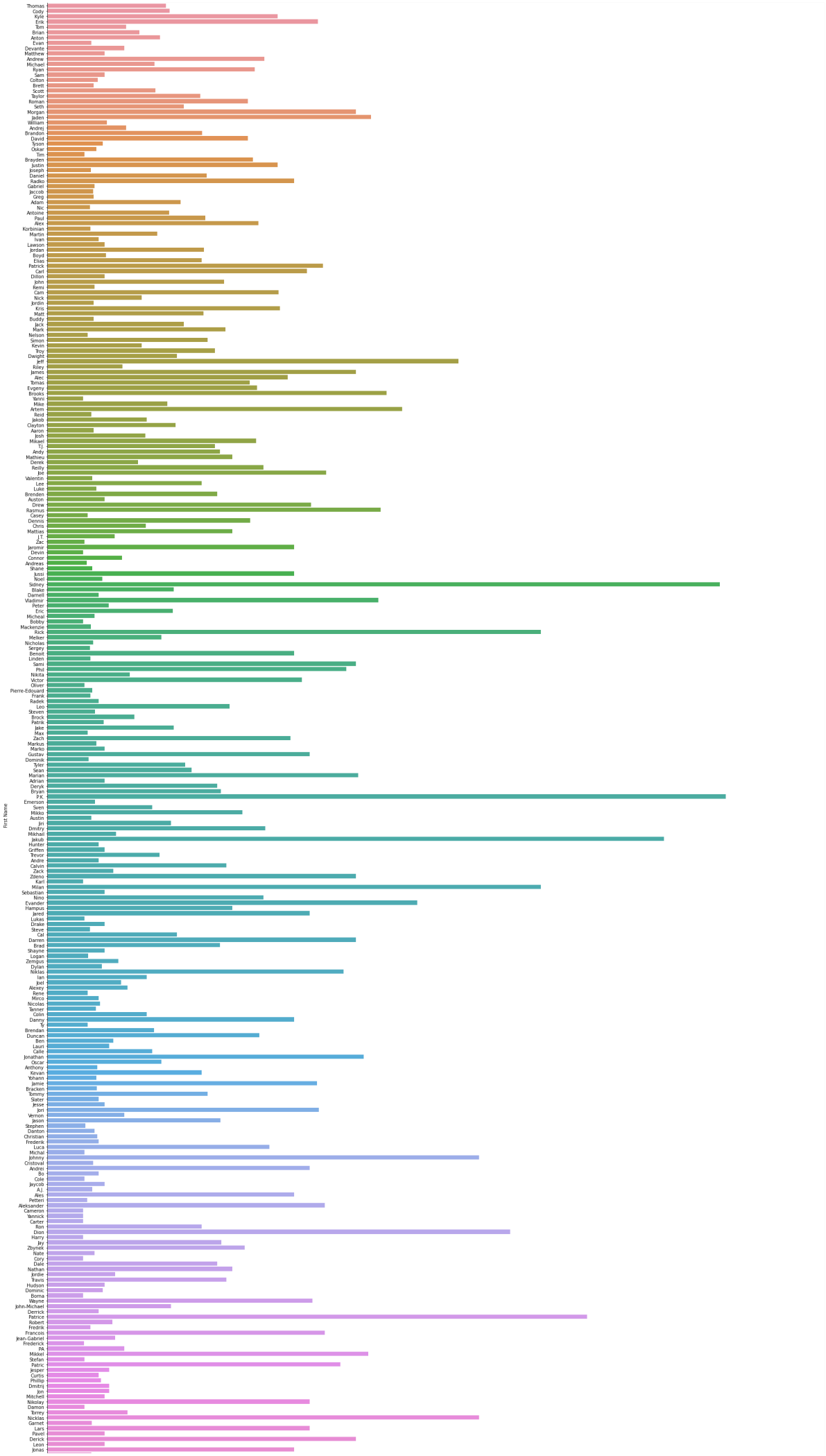

First Name
Hand
L
R

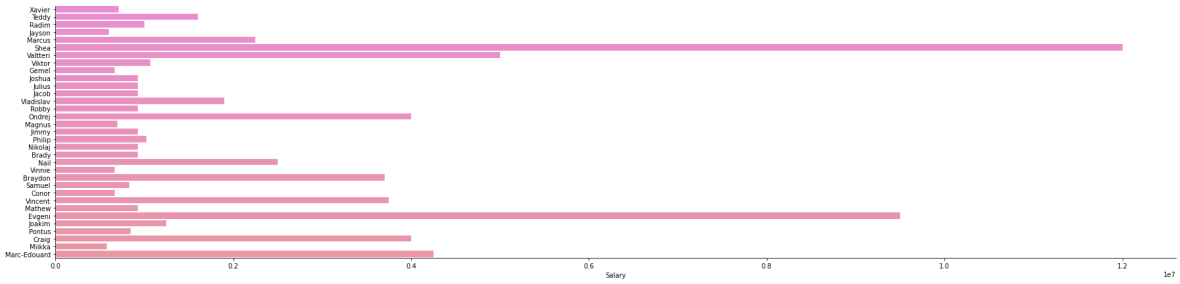


Maximum number of goals are scored by left handed player SIDNEY and by right handed player Patrik

In [29]:
```
plt.figure(figsize=(100,100))
sns.catplot(x='Salary',y='First Name', kind = 'bar', data = data , height=50,aspec
```

Out[29]:
```
<seaborn.axisgrid.FacetGrid at 0x225fb5091f0>

<Figure size 7200x7200 with 0 Axes>
```


First Name
Thomas
Cody
Kyle
Erik
Tom
Brian
Anton
Evan
Devante
Matthew
Andrew
Michael
Ryan
Sam
Colton
Brett
Scott
Taylor
Roman
Seth
Morgan
Jaden
William
Andrei
Brandon
David
Tyson
Oskar
Tim
Brayden
Justin
Joseph
Daniel
Radko
Gabriel
Jaccob
Greg
Adam
Nic
Antoine
Paul
Alex
Korbinian
Martin
Ivan
Lawson
Jordan
Boyd
Elias
Patrick
Carl
Dillon
John
Remi
Cam
Nick
Jordin
Kris
Matt
Buddy
Jack
Mark
Nelson
Simon
Kevin
Troy
Dwight
Jeff
Riley
James
Alec
Tomas
Evgeny
Brooks
Yanni
Mike
Artem
Reid
Jakob
Clayton
Aaron
Josh
Mikael
TJ
Andy
Mathieu
Derek
Reilly
Joe
Valentin
Lee
Luke
Brenden
Auston
Drew
Rasmus
Casey
Dennis
Chris
Mattias
J.T
Zac
Jeromie
Devin
Connor
Andreas
Shane
Jussi
Noel
Sidney
Blake
Darnell
Vladimir
Peter
Eric
Micheal
Bobby
Mackenzie
Rick
Melker
Nicholas
Sergey
Benoit
Linden
Sami
Phil
Nikita
Victor
Oliver
Pierre-Edouard
Frank
Radek
Leo
Steven
Brock
Patrik
Jake
Max
Zach
Markus
Marko
Gustav
Dominik
Tyler
Sean
Marian
Adrian
Deryk
Bryan
P.K.
Emerson
Sven
Mikko
Austin
Jiri
Dmitry
Mikhail
Jakub
Hunter
Griffen
Trevor
Andre
Calvin
Zack
Zdeno
Kari
Milan
Sebastian
Nino
Evander
Hampus
Jared
Lukas
Drake
Steve
Cal
Darren
Brad
Shayne
Logan
Zemgus
Dylan
Niklas
Ian
Joel
Alexey
Rene
Mirco
Nicolas
Tanner
Colin
Danny
Ty
Brendan
Duncan
Ben
Laun
Calle
Jonathan
Oscar
Anthony
Kevan
Yohann
Jamie
Bracken
Tommy
Slater
Jesse
Jori
Vernon
Jason
Stephen
Danton
Christian
Frederik
Luca
Michal
Johnny
Cristoval
Andrej
Bo
Cole
Jaycob
A.J.
Ales
Petteri
Aleksander
Cameron
Yannick
Carter
Ron
Dion
Harry
Jay
Zbynek
Nate
Cory
Dalé
Nathan
Jordie
Travis
Hudson
Dominic
Borna
Wayne
John-Michael
Derrick
Patrice
Robert
Fredrik
Francois
Jean-Gabriel
Frederick
PA
Mikkel
Stefan
Patric
Jesper
Curtis
Phillip
Dmitrij
Jon
Mitchell
Nikolay
Damon
Torrey
Nicklas
Garnet
Lars
Pavel
Derick
Leon
Jonas



Shea holds the position for highest Salary

Data Integration

In [30]:
```
# Separating Categorical and Numerical variables

num_features = [column_name for column_name in data.columns if data[column_name].d
cat_features = [column_name for column_name in data.columns if data[column_name].d
print('Numerical :',num_features)
print('Categorical :',cat_features)
columns=num_features+cat_features
columns
```

```
Numerical : ['Salary', 'Ht', 'Wt', 'DftYr', 'DftRd', 'Ovrl', 'GP', 'G', 'A', 'A1',
'A2', 'PTS', '+/-', 'E+/-', 'PIM', 'Shifts', 'TOI', 'TOIX', 'TOI/GP', 'TOI/GP.1',
'TOI%', 'IPP%', 'SH%', 'SV%', 'PDO', 'F/60', 'A/60', 'Pct%', 'Diff', 'Diff/60', 'i
CF', 'iCF.1', 'iFF', 'iSF', 'iSF.1', 'iSF.2', 'ixG', 'iSCF', 'iRB', 'iRS', 'iDS',
'sDist', 'sDist.1', 'Pass', 'iHF', 'iHF.1', 'iHA', 'iHDf', 'iMiss', 'iGVA', 'iTK
A', 'iBLK', 'iGVA.1', 'iTKA.1', 'iBLK.1', 'BLK%', 'iFOW', 'iFOL', 'iFOW.1', 'iFOL.
1', 'FO%', '%FOT', 'dzFOW', 'dzFOL', 'nzFOW', 'nzFOL', 'ozFOW', 'ozFOL', 'FOW.Up',
'FOL.Up', 'FOW.Down', 'FOL.Down', 'FOW.Close', 'FOL.Close', 'OTG', '1G', 'GWG', 'E
NG', 'PSG', 'PSA', 'G.Bkhd', 'G.Dflct', 'G.Slap', 'G.Snap', 'G.Tip', 'G.Wrap', 'G.
Wrst', 'CBar ', 'Post', 'Over', 'Wide', 'S.Bkhd', 'S.Dflct', 'S.Slap', 'S.Snap',
'S.Tip', 'S.Wrap', 'S.Wrst', 'iPenT', 'iPenD', 'iPENT', 'iPEND', 'iPenDf', 'NPD',
'Min', 'Maj', 'Match', 'Misc', 'Game', 'CF', 'CA', 'FF', 'FA', 'SF', 'SA', 'xGF',
'xGA', 'SCF', 'SCA', 'GF', 'GA', 'RBF', 'RBA', 'RSF', 'RSA', 'DSF', 'DSA', 'FOW',
'FOL', 'HF', 'HA', 'GVA', 'TKA', 'PENT', 'PEND', 'OPS', 'DPS', 'PS', 'OTOI', 'Gri
t', 'DAP', 'Pace', 'GS', 'GS/G']
Categorical : ['Born', 'City', 'Pr/St', 'Cntry', 'Nat', 'Hand', 'Last Name', 'Firs
t Name', 'Position', 'Team']
```

Out[30]:
```
['Salary',
 'Ht',
 'Wt',
 'DftYr',
 'DftRd',
 'Ovrl',
 'GP',
 'G',
 'A',
 'A1',
 'A2',
 'PTS',
 '+/-',
 'E+/-',
 'PIM',
 'Shifts',
 'TOI',
 'TOIX',
 'TOI/GP',
 'TOI/GP.1',
 'TOI%',
 'IPP%',
 'SH%',
 'SV%',
 'PDO',
 'F/60',
 'A/60',
 'Pct%',
 'Diff',
 'Diff/60',
 'iCF',
 'iCF.1',
 'iFF',
 'iSF',
 'iSF.1',
 'iSF.2',
 'ixG',
 'iSCF',
 'iRB',
 'iRS',
 'iDS',
 'sDist',
 'sDist.1',
 'Pass',
 'iHF',
 'iHF.1',
 'iHA',
 'iHDf',
 'iMiss',
 'iGVA',
 'iTKA',
 'iBLK',
 'iGVA.1',
 'iTKA.1',
 'iBLK.1',
 'BLK%',
 'iFOW',
 'iFOL',
 'iFOW.1',
 'iFOL.1',
 'FO%',
 '%FOT',
 'dzFOW',
 'dzFOL',
```

```
'nzFOW',
'nzFOL',
'ozFOW',
'ozFOL',
'FOW.Up',
'FOL.Up',
'FOW.Down',
'FOL.Down',
'FOW.Close',
'FOL.Close',
'OTG',
'1G',
'GWG',
'ENG',
'PSG',
'PSA',
'G.Bkhd',
'G.Dflct',
'G.Slap',
'G.Snap',
'G.Tip',
'G.Wrap',
'G.Wrst',
'CBar ',
'Post',
'Over',
'Wide',
'S.Bkhd',
'S.Dflct',
'S.Slap',
'S.Snap',
'S.Tip',
'S.Wrap',
'S.Wrst',
'iPenT',
'iPenD',
'iPENT',
'iPEND',
'iPenDf',
'NPD',
'Min',
'Maj',
'Match',
'Misc',
'Game',
'CF',
'CA',
'FF',
'FA',
'SF',
'SA',
'xGF',
'xGA',
'SCF',
'SCA',
'GF',
'GA',
'RBF',
'RBA',
'RSF',
'RSA',
'DSF',
'DSA',
'FOW',
```

```
'FOL',
'HF',
'HA',
'GVA',
'TKA',
'PENT',
'PEND',
'OPS',
'DPS',
'PS',
'OTOI',
'Grit',
'DAP',
'Pace',
'GS',
'GS/G',
'Born',
'City',
'Pr/St',
'Cntry',
'Nat',
'Hand',
'Last Name',
'First Name',
'Position',
'Team']
```

Correlation Analysis for categorical data

Chi Square test

In [31]:
```
import scipy.stats as stats
import statsmodels.api as sm
for i in range(0,10):
    crosstab= pd.crosstab(data[cat_features[i]], [data["Salary"]])
    x= stats.chi2_contingency(crosstab)
    x=pd.DataFrame(x)
    (chi2,p,dof)=(x.iloc[0],x.iloc[1],x.iloc[2])
    y=pd.DataFrame(data[cat_features])

    print(y.columns[i])
    print('chi2    :' ,chi2)
    print('p value:',p)
    print('Degree of freedom  :',dof)
    print('****************************************')
```

```
Born
chi2    : 0    78273.512251
Name: 0, dtype: object
p value: 0    0.320722
Name: 1, dtype: object
Degree of freedom  : 0    78090
Name: 2, dtype: object
****************************************
City
chi2    : 0    50357.491704
Name: 0, dtype: object
p value: 0    0.929901
Name: 1, dtype: object
Degree of freedom  : 0    50827
Name: 2, dtype: object
****************************************
Pr/St
chi2    : 0    6219.072896
Name: 0, dtype: object
p value: 0    0.0
Name: 1, dtype: object
Degree of freedom  : 0    4932
Name: 2, dtype: object
****************************************
Cntry
chi2    : 0    3564.752715
Name: 0, dtype: object
p value: 0    0.0
Name: 1, dtype: object
Degree of freedom  : 0    2329
Name: 2, dtype: object
****************************************
Nat
chi2    : 0    2491.852217
Name: 0, dtype: object
p value: 0    0.0
Name: 1, dtype: object
Degree of freedom  : 0    2055
Name: 2, dtype: object
****************************************
Hand
chi2    : 0    157.207477
Name: 0, dtype: object
p value: 0    0.114051
Name: 1, dtype: object
Degree of freedom  : 0    137
Name: 2, dtype: object
****************************************
Last Name
chi2    : 0    78574.677851
Name: 0, dtype: object
p value: 0    0.027517
Name: 1, dtype: object
Degree of freedom  : 0    77816
Name: 2, dtype: object
****************************************
First Name
chi2    : 0    42736.881703
Name: 0, dtype: object
p value: 0    0.000537
Name: 1, dtype: object
Degree of freedom  : 0    41785
Name: 2, dtype: object
****************************************
```

```
Position
chi2    : 0    1998.836489
Name: 0, dtype: object
p value: 0    1.0
Name: 1, dtype: object
Degree of freedom  : 0    2329
Name: 2, dtype: object
**************************************
Team
chi2    : 0    8832.58636
Name: 0, dtype: object
p value: 0    0.941146
Name: 1, dtype: object
Degree of freedom  : 0    9042
Name: 2, dtype: object
**************************************
```

The dependent variables with the target variable Salary are

- Pr/St
- Country
- Nat
- Last Name
- First Name

Correlation Analysis for Numeric data

Checking through Correlation heatmap

In [32]:
```
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
pd.set_option('display.max_colwidth', -1)
```

In [33]:
```
correlation_mat = data.corr()

corr_pairs = correlation_mat.unstack()

print(corr_pairs)
```

```
Salary  Salary    1.000000
        Ht        0.087047
        Wt        0.182023
        DftYr    -0.469829
        DftRd    -0.167339
                    ...
GS/G    Grit      0.230489
        DAP       0.039756
        Pace      0.341186
        GS        0.881550
        GS/G      1.000000
Length: 20736, dtype: float64
```

In [34]:
```
sorted_pairs = corr_pairs.sort_values(kind="quicksort")

print(sorted_pairs)
```

```
A/60      SV%       -0.883473
SV%       A/60      -0.883473
Diff/60   A/60      -0.774845
A/60      Diff/60   -0.774845
PDO       A/60      -0.690104
                       ...
S.Slap    S.Slap     1.000000
S.Dflct   S.Dflct    1.000000
S.Bkhd    S.Bkhd     1.000000
S.Wrst    S.Wrst     1.000000
GS/G      GS/G       1.000000
Length: 20736, dtype: float64
```

In [35]:
```
negative_pairs = sorted_pairs[sorted_pairs < -0.5]

print(negative_pairs)
```

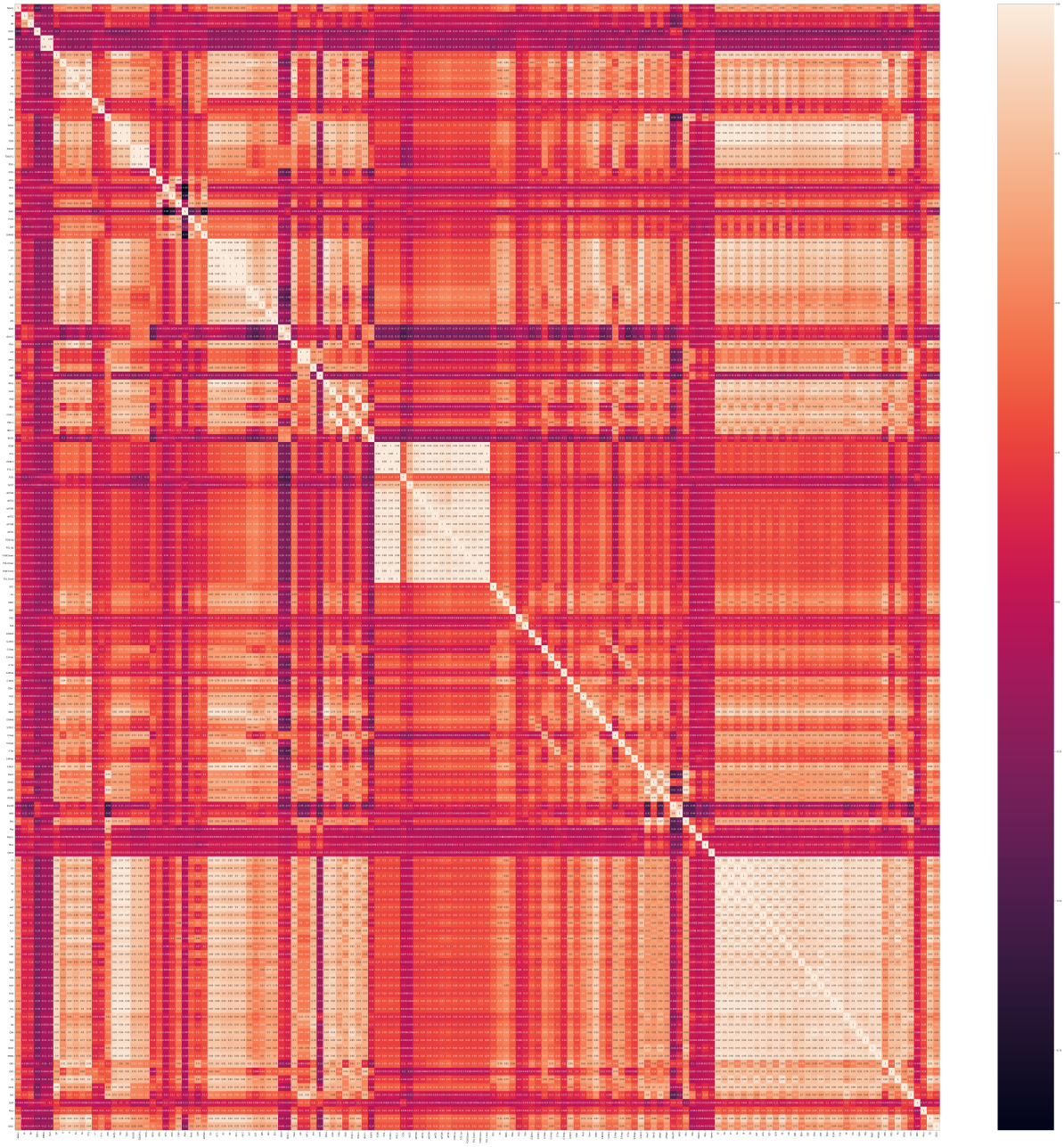
```
A/60      SV%       -0.883473
SV%       A/60      -0.883473
Diff/60   A/60      -0.774845
A/60      Diff/60   -0.774845
PDO       A/60      -0.690104
A/60      PDO       -0.690104
PIM       iPenDf    -0.557674
iPenDf    PIM       -0.557674
sDist.1   FO%       -0.530661
FO%       sDist.1   -0.530661
iPenT     iPenDf    -0.520910
iPenDf    iPenT     -0.520910
iPENT     iPenDf    -0.508297
iPenDf    iPENT     -0.508297
dtype: float64
```

In [36]:
```
strong_pairs = sorted_pairs[(sorted_pairs) > 0.7]

print(strong_pairs)
```

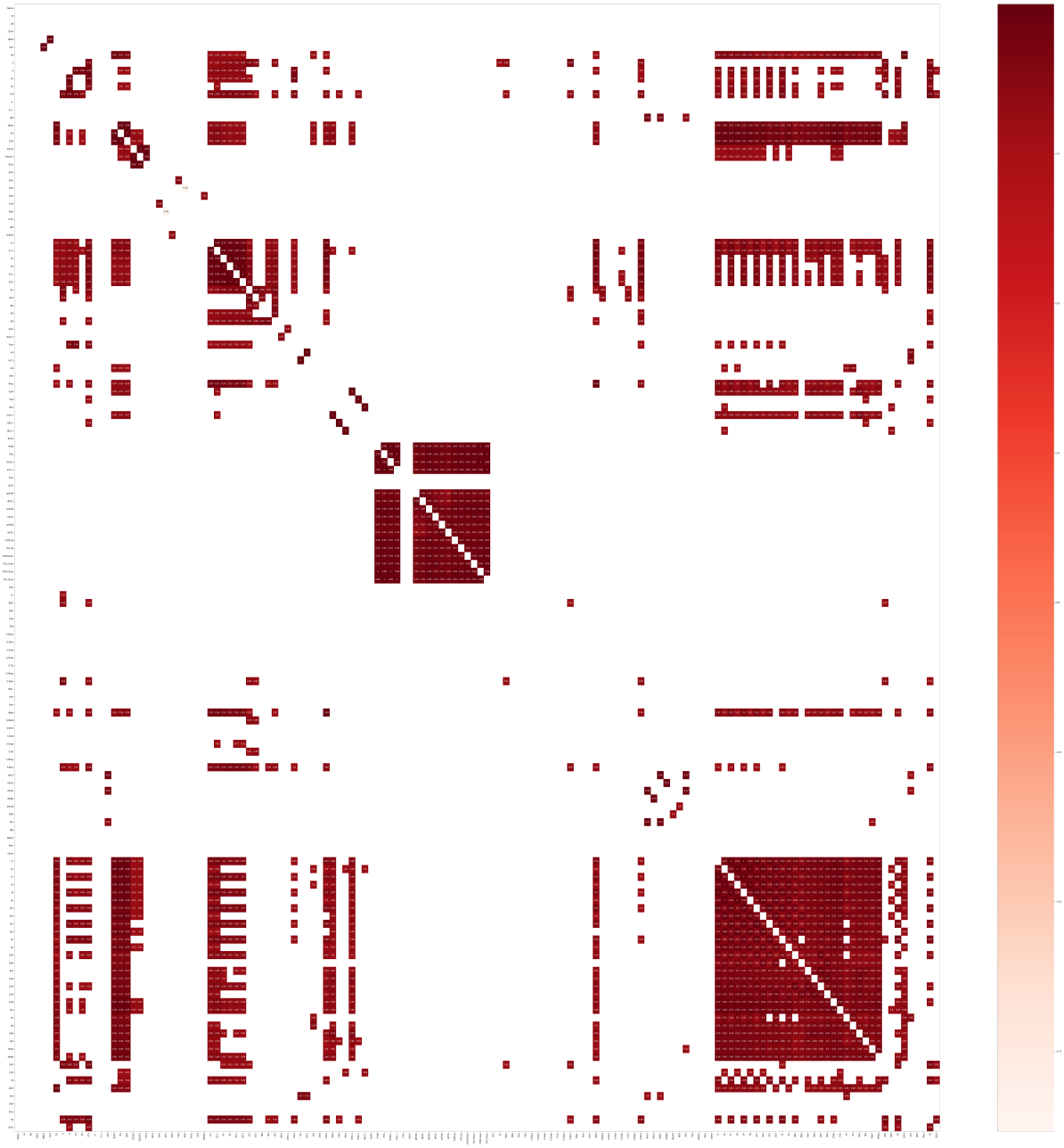
```
HA        TOI%       0.700128
TOI%      HA         0.700128
G.Tip     iSCF       0.700406
iSCF      G.Tip      0.700406
G         CF         0.700426
                        ...
S.Slap    S.Slap     1.000000
S.Dflct   S.Dflct    1.000000
S.Bkhd    S.Bkhd     1.000000
S.Wrst    S.Wrst     1.000000
GS/G      GS/G       1.000000
Length: 3982, dtype: float64
```

In [37]:
```
corr = data.corr()
fig, ax = plt.subplots()
fig.set_size_inches(100,100)
sns.heatmap(corr,annot=True)
```

Out[37]:
```
<AxesSubplot:>
```



In [38]:
```
dataCorr = data.corr()
filteredDf = dataCorr[((dataCorr >= .80) | (dataCorr <= -.80)) & (dataCorr !=1.000)
plt.figure(figsize=(100,100))
sns.heatmap(filteredDf, annot=True, cmap="Reds")
plt.show()
```



In [39]:
```
def correlation(data, threshold=None):
    # Set of all names of correlated columns
    col_corr = set()
    corr_mat = data.corr()
    for i in range(len(corr_mat.columns)):
        for j in range(i):
            if (abs(corr_mat.iloc[i,j]) > threshold):
                colname = corr_mat.columns[i]
                col_corr.add(colname)
    return col_corr
```

In [40]:
```
correlated_features = correlation(data=data, threshold=0.8)
correlated_features
```

Out[40]:
```
{'1G',
 'A/60',
 'A1',
 'A2',
 'CA',
 'CF',
 'DPS',
 'DSA',
 'DSF',
 'Diff/60',
 'F/60',
 'FA',
 'FF',
 'FOL',
 'FOL.Close',
 'FOL.Down',
 'FOL.Up',
 'FOW',
 'FOW.Close',
 'FOW.Down',
 'FOW.Up',
 'G.Wrst',
 'GA',
 'GF',
 'GS',
 'GS/G',
 'GVA',
 'GWG',
 'Grit',
 'HA',
 'HF',
 'Min',
 'NPD',
 'OPS',
 'OTOI',
 'Ovrl',
 'PEND',
 'PENT',
 'PS',
 'PTS',
 'Pass',
 'RBA',
 'RBF',
 'RSA',
 'RSF',
 'S.Bkhd',
 'S.Snap',
 'S.Tip',
 'S.Wrst',
 'SA',
 'SCA',
 'SCF',
 'SF',
 'Shifts',
 'TKA',
 'TOI',
 'TOI%',
 'TOI/GP',
 'TOI/GP.1',
 'TOIX',
 'Wide',
 'dzFOL',
 'dzFOW',
 'iBLK.1',
```

```
 'iCF',
 'iCF.1',
 'iDS',
 'iFF',
 'iFOL',
 'iFOL.1',
 'iFOW.1',
 'iGVA',
 'iGVA.1',
 'iHA',
 'iHF.1',
 'iMiss',
 'iPEND',
 'iPENT',
 'iPenT',
 'iRB',
 'iRS',
 'iSCF',
 'iSF',
 'iSF.1',
 'iSF.2',
 'iTKA',
 'iTKA.1',
 'ixG',
 'nzFOL',
 'nzFOW',
 'ozFOL',
 'ozFOW',
 'sDist.1',
 'xGA',
 'xGF'}
```

Removing duplicates

In [41]:
```
duplicate = data.duplicated()
print(duplicate.sum())   # This code will give sum of or  total number of duplicate
data[duplicate]   #  This command shows all the duplicate rows.
```

```
0
```

Out[41]:

	Salary	Born	City	Pr/St	Cntry	Nat	Ht	Wt	DftYr	DftRd	Ovrl	Hand	Last Name	First Name	Posi

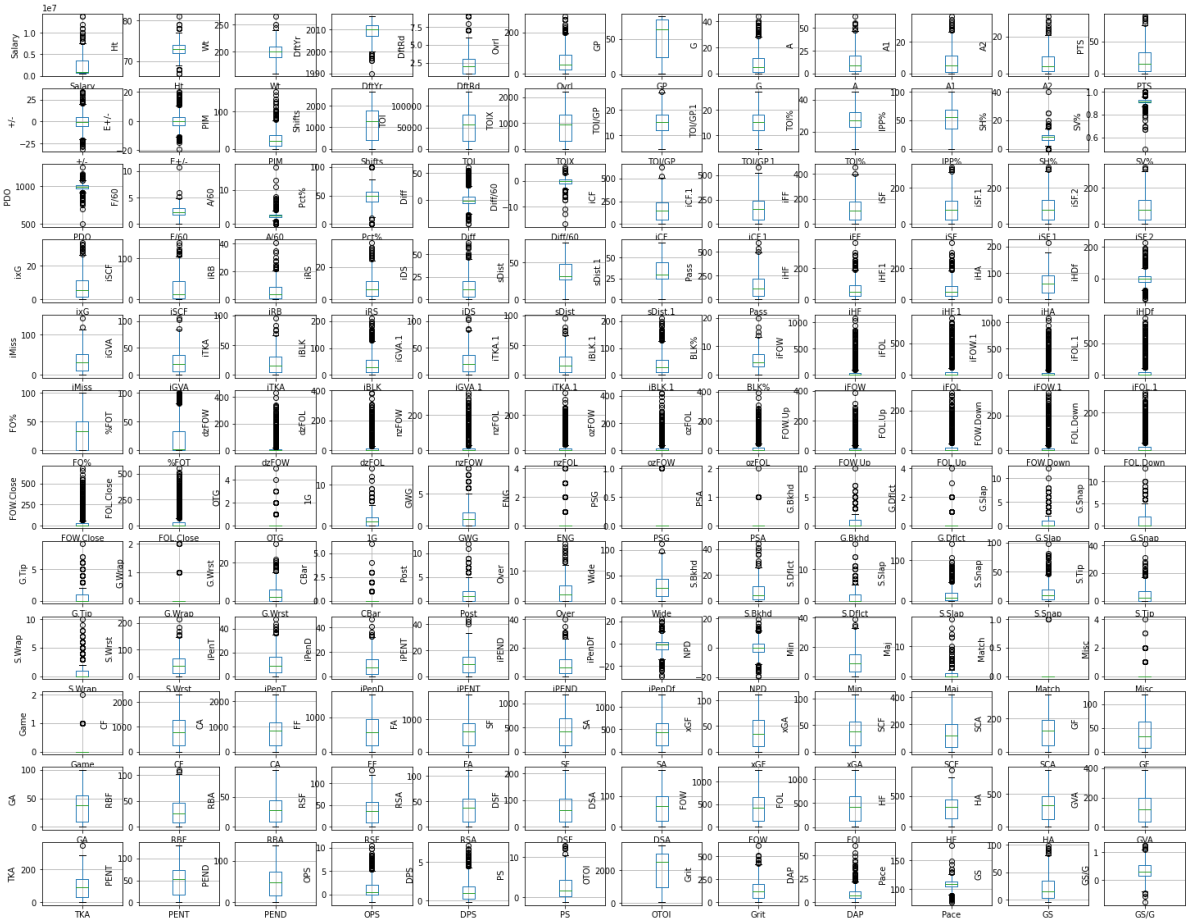
In [42]:
```
# data.drop_duplicates(inplace = True)   # dropping duplicate rows.
```

In [43]:
```
# dp1 = data.duplicated()   # checking for duplicates one more time.
# dp1.sum()
```

Dealing with Outliers

In [44]:
```
numerical=['Salary', 'Ht', 'Wt', 'DftYr', 'DftRd', 'Ovrl', 'GP', 'G', 'A', 'A1', 'A
print(len(numerical))
plt.figure(figsize=(25,20))
for i in range(len(numerical)):
    plt.subplot(12,12, i+1)
    fig = data.boxplot(column=numerical[i])
    fig.set_title('')
    fig.set_ylabel(numerical[i])
```

144

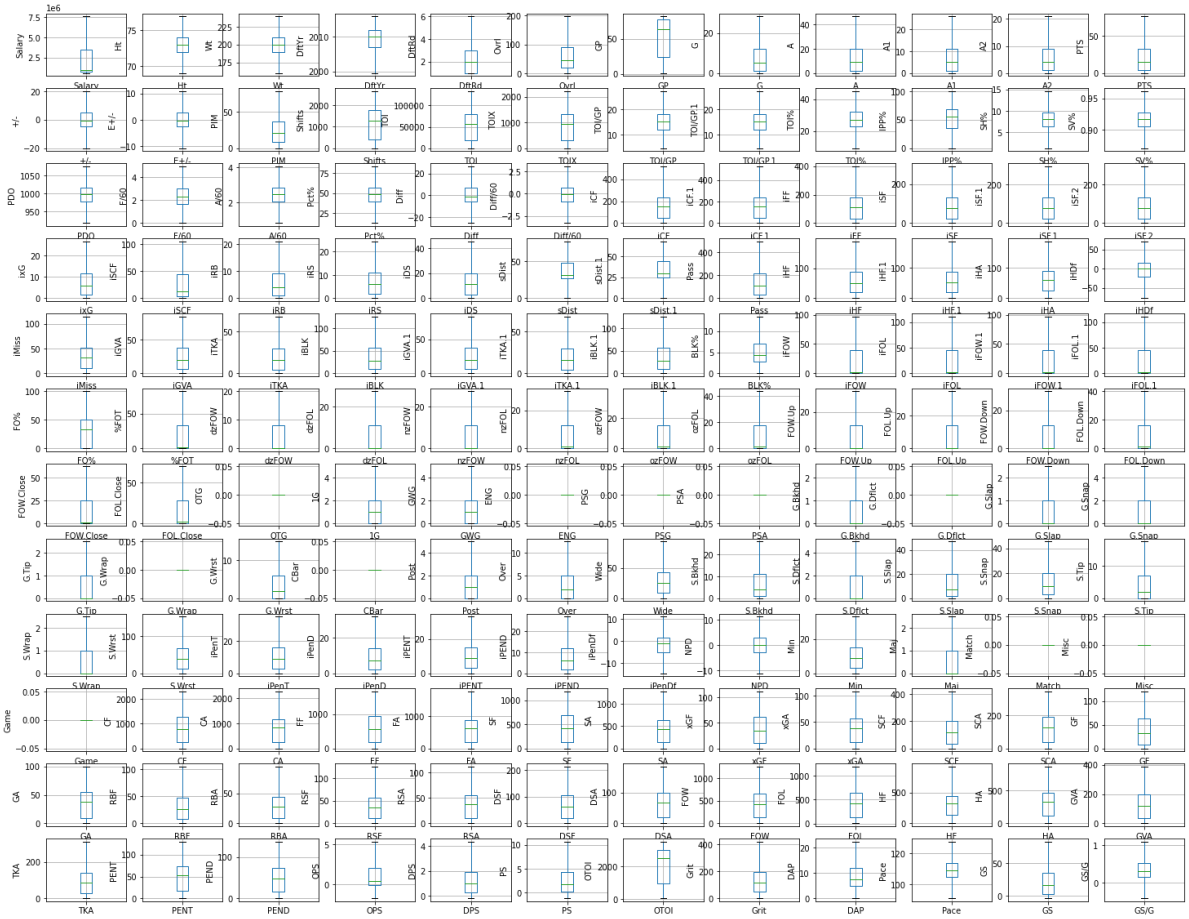


Using IQR capping to smoothen the dataset

In [45]:
```
def iqr_capping(data, cols, factor):

    for col in cols:

        q1 = data[col].quantile(0.25)
        q3 = data[col].quantile(0.75)

        iqr = q3 - q1

        upper_whisker = q3 + (factor*iqr)
        lower_whisker = q1 - (factor*iqr)

        data[col] = np.where(data[col]>upper_whisker, upper_whisker,
                np.where(data[col]<lower_whisker, lower_whisker, data[col]))
```

In [46]:
```
iqr_capping(data, numerical, 1.5)
```

In [47]:
```
plt.figure(figsize=(25,20))
for i in range(len(numerical)):
    plt.subplot(12, 12, i+1)
    fig = data.boxplot(column=numerical[i])
    fig.set_title('')
    fig.set_ylabel(numerical[i])
```



In [48]:
```
data.shape
```

Out[48]:
```
(607, 154)
```

The resultant dataset is now smooth

Data Transformation

Min-max Normalization

In [49]:
```
data_norm=data.copy()
```

In [50]:
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data_norm = scaler.fit_transform(data_norm[numerical])
```

In [51]:
```
data_nr=pd.DataFrame(data_norm,columns=numerical)
data_nr
```

Out[51]:

	Salary	Ht	Wt	DftYr	DftRd	Ovrl	GP	G	A	A1	
0	0.049566	0.625	0.3750	0.939394	0.0	0.086404	0.000000	0.000000	0.000000	0.000000	(
1	0.237210	0.625	0.5875	0.757576	0.0	0.071156	0.962963	0.070175	0.319149	0.230769	(
2	1.000000	0.375	0.7250	0.393939	0.0	0.030496	0.790123	0.666667	0.553191	0.500000	(
3	0.414233	1.000	0.7500	0.636364	0.0	0.010165	0.358025	0.035088	0.106383	0.192308	(
4	0.166401	0.875	0.7125	0.757576	0.0	0.076239	1.000000	0.245614	0.255319	0.153846	(
...	...	...	...	...	...	...	...	...	...	...	
602	0.003540	0.500	0.5125	0.696970	0.4	0.411690	0.851852	0.210526	0.234043	0.230769	(
603	1.000000	0.625	0.5750	0.212121	0.0	0.030496	1.000000	0.315789	0.659574	0.384615	
604	0.520446	0.500	0.5625	0.333333	0.2	0.172808	0.913580	0.210526	0.468085	0.538462	(
605	0.909896	0.250	0.4500	0.090909	0.4	0.355781	0.950617	0.350877	0.382979	0.346154	(
606	0.049566	1.000	0.8250	0.757576	0.4	0.432020	0.987654	0.140351	0.659574	0.653846	(

607 rows × 144 columns

Z-score Normalization

In [52]:
```
data_st=data.copy()
```

In [53]:
```
x=data_st.drop(['Born', 'City', 'Pr/St', 'Cntry', 'Nat', 'Hand', 'Last Name', 'Firs
```

In [54]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
standard_x=scaler.fit_transform(x)
```

In [55]:
```
standard_x
```

Out[55]:
```
array([[ 0.52151896, -0.7212315 ,  1.51869082, ...,  2.48365088,
        -1.0423807 , -2.5045571 ],
       [ 0.52151896,  0.4292381 ,  0.75504703, ...,  0.47422034,
        -0.37442802, -0.5579045 ],
       [-0.49310287,  1.17365961, -0.77224055, ...,  0.77678599,
         0.67126341,  0.7978    ],
       ...,
       [ 0.01420805,  0.29388874, -1.02678847, ..., -0.0125157 ,
         0.4731809 ,  0.31113684],
       [-1.00041378, -0.31518341, -2.04498019, ...,  0.56630554,
         0.75878825,  0.55446842],
       [ 2.04345171,  1.71505707,  0.75504703, ..., -0.64395705,
         1.08585473,  0.76303834]])
```

In [56]:
```
standard_data1_x=pd.DataFrame(standard_x)
```

In [57]:
```
standard_data1_x
```

Out[57]:

	0	1	2	3	4	5	6	7	
0	0.521519	-0.721232	1.518691	-0.918347	-0.800156	-1.753129	-0.942658	-1.048642	-0.9995
1	0.521519	0.429238	0.755047	-0.918347	-0.854330	0.938735	-0.693526	0.162557	-0.1626
2	-0.493103	1.173660	-0.772241	-0.918347	-0.998792	0.455580	1.424094	1.050770	0.8136
3	2.043452	1.309009	0.245951	-0.918347	-1.071023	-0.752307	-0.818092	-0.644909	-0.3021
4	1.536141	1.105985	0.755047	-0.918347	-0.836272	1.042269	-0.070697	-0.079682	-0.4416
...	...	...	...	...	...	...	...	...	
602	0.014208	0.023190	0.500499	0.286921	0.355540	0.628136	-0.195263	-0.160429	-0.1626
603	0.521519	0.361563	-1.535884	-0.918347	-0.998792	1.042269	0.178435	1.454504	0.3952
604	0.014208	0.293889	-1.026788	-0.315713	-0.493174	0.800691	-0.195263	0.727784	0.9531
605	-1.000414	-0.315183	-2.044980	0.286921	0.156905	0.904224	0.303001	0.404797	0.2557
606	2.043452	1.715057	0.755047	0.286921	0.427771	1.007758	-0.444395	1.454504	1.3716

607 rows × 143 columns

Data Reduction

Principal Component Analysis

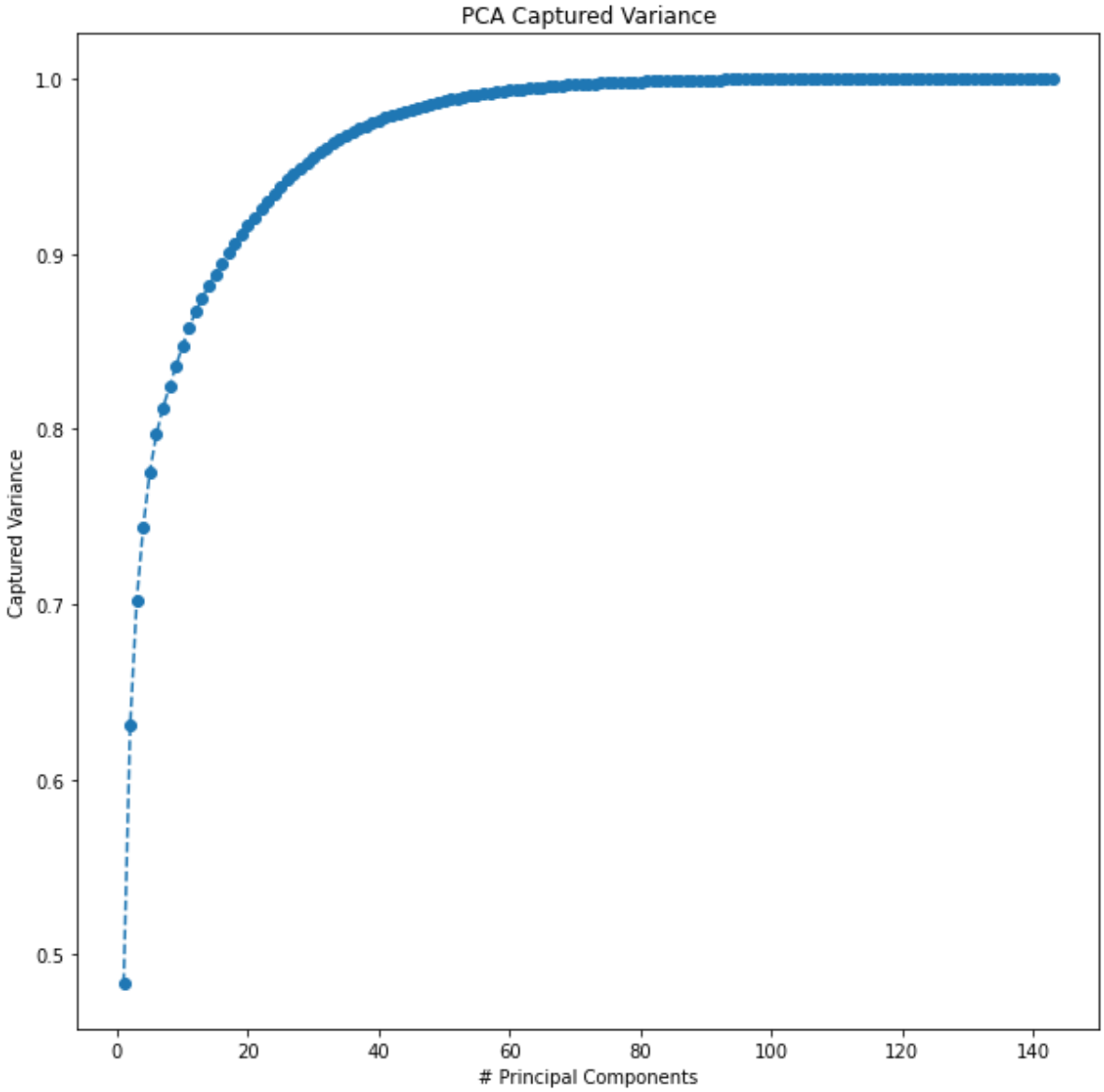
In [58]:
```
from sklearn.decomposition import PCA
pca=PCA().fit(standard_data1_x)
pca_df=pca.transform(standard_data1_x)
pca_df.shape
```

Out[58]:
```
(607, 143)
```

In [59]:
```
pca.explained_variance_ratio_
```

Out[59]:
```
array([4.83588257e-01, 1.47017429e-01, 7.16801765e-02, 4.20003978e-02,
       3.10728016e-02, 2.14390718e-02, 1.51271495e-02, 1.22017152e-02,
       1.18291786e-02, 1.13060392e-02, 1.08453492e-02, 8.69987514e-03,
       7.84228046e-03, 7.08558924e-03, 6.44181461e-03, 6.21993056e-03,
       5.84226508e-03, 5.60224543e-03, 5.31796692e-03, 5.02059788e-03,
       4.77880678e-03, 4.49376969e-03, 4.38652460e-03, 4.23291884e-03,
       4.11072147e-03, 3.77904816e-03, 3.67483682e-03, 3.30669119e-03,
       3.08966520e-03, 2.96250727e-03, 2.83115299e-03, 2.70915433e-03,
       2.37847228e-03, 2.31354412e-03, 2.15730825e-03, 2.01343347e-03,
       1.82443409e-03, 1.75582652e-03, 1.61336150e-03, 1.48150121e-03,
       1.43294998e-03, 1.34981442e-03, 1.30273214e-03, 1.20934031e-03,
       1.15270863e-03, 1.07750978e-03, 1.00699717e-03, 9.24168407e-04,
       9.11306085e-04, 8.63007827e-04, 8.31494102e-04, 7.25166592e-04,
       6.96885006e-04, 6.79319240e-04, 6.16748686e-04, 5.77877333e-04,
       5.45489016e-04, 5.16814210e-04, 4.41345990e-04, 4.21463323e-04,
       4.04536401e-04, 3.75028939e-04, 3.55035610e-04, 3.38285695e-04,
       3.16419005e-04, 3.01724361e-04, 2.84544165e-04, 2.63664296e-04,
       2.57469962e-04, 2.50187228e-04, 2.32038763e-04, 2.14411109e-04,
       2.01885489e-04, 1.88162610e-04, 1.77564618e-04, 1.62586651e-04,
       1.61734030e-04, 1.53574537e-04, 1.37946517e-04, 1.33684255e-04,
       1.20120312e-04, 1.18047190e-04, 1.04196620e-04, 1.03202341e-04,
       9.75230278e-05, 9.05074767e-05, 8.83457573e-05, 8.42593378e-05,
       8.18663287e-05, 7.68178095e-05, 7.28304709e-05, 6.85198959e-05,
       6.58972777e-05, 6.19091758e-05, 5.40562810e-05, 5.02933220e-05,
       4.45957778e-05, 4.20738579e-05, 3.87702053e-05, 3.56350154e-05,
       3.01233276e-05, 2.71820993e-05, 2.14846532e-05, 2.06891628e-05,
       1.83555938e-05, 1.75021084e-05, 1.48576332e-05, 1.20724524e-05,
       1.02404583e-05, 1.01353984e-05, 9.28946491e-06, 6.71919081e-06,
       5.28681698e-06, 4.88208835e-06, 4.32705955e-06, 3.75489675e-06,
       3.51398485e-06, 3.17963364e-06, 2.99557930e-06, 1.70535019e-06,
       1.52266838e-06, 1.04602650e-06, 5.39605130e-07, 4.20694193e-07,
       3.26557992e-07, 2.90970459e-07, 2.06930394e-07, 1.64776535e-07,
       1.07707699e-07, 8.86102916e-08, 7.21821213e-08, 1.72504546e-08,
       1.98420311e-33, 1.98420311e-33, 1.98420311e-33, 1.98420311e-33,
       1.98420311e-33, 1.98420311e-33, 1.98420311e-33, 1.98420311e-33,
       1.98420311e-33, 1.98420311e-33, 1.98420311e-33, 1.98420311e-33])
```

In [60]:
```
plt.figure(figsize=(10,10))
plt.plot(range(1, 144), pca.explained_variance_ratio_.cumsum(), marker='o', linesty
plt.title('PCA Captured Variance')
plt.xlabel('# Principal Components')
plt.ylabel('Captured Variance')
plt.show()
```



The top 17 principal components shows the highest variability of around 90% in our dataset

In [61]:
```
#so now we'll run PCA again with our chosen number of principal components:
pca = PCA(n_components=17)
pca.fit(standard_data1_x)
pca.components_
```

Out[61]:
```
array([[ 0.01046276,  0.01310676, -0.03208805, ...,  0.05239412,
         0.1137606 ,  0.09574485],
       [-0.04535933, -0.04664244,  0.01026242, ..., -0.004862  ,
         0.01869737,  0.021278  ],
       [-0.07843328, -0.08366334,  0.06073338, ...,  0.05041264,
         0.1035408 ,  0.13542676],
       ...,
       [ 0.10150578,  0.15702569, -0.10176643, ...,  0.37832391,
        -0.03155705, -0.09356229],
       [ 0.13515602,  0.20273927, -0.22699873, ...,  0.14498385,
         0.01704527,  0.18078136],
       [-0.06982451, -0.14553001,  0.2234011 , ...,  0.13395363,
         0.00551114, -0.03360538]])
```

In [62]:
```
#now we make a new dataframe with these components
#and set the columns to the original columns of our data
data_pca = pd.DataFrame(data = pca.components_,
                        columns = standard_data1_x.columns.values,
```

```
                                index = ['Component 1', 'Component 2', 'Component 3', '
data_pca
```

Out[62]:

	0	1	2	3	4	5	6	7
Component 1	0.010463	0.013107	-0.032088	-0.021631	-0.022208	0.113553	0.101292	0.112235
Component 2	-0.045359	-0.046642	0.010262	-0.011995	-0.010044	-0.007809	0.067697	0.011066
Component 3	-0.078433	-0.083663	0.060733	-0.016134	-0.019203	-0.069834	0.116094	0.070503
Component 4	0.044532	0.106917	-0.041723	0.040324	0.042671	0.059702	0.066516	-0.061209
Component 5	0.064722	0.082631	-0.032756	0.019062	0.023504	0.003275	-0.043510	0.009127
Component 6	-0.239706	-0.190538	-0.029599	0.317930	0.322499	0.102345	-0.052977	-0.064143
Component 7	-0.250489	-0.224031	-0.234169	0.475073	0.473133	-0.039210	0.016455	0.048429
Component 8	0.259064	0.321731	-0.105422	0.127397	0.118154	-0.062327	0.013385	0.007898
Component 9	-0.029522	-0.028994	-0.094098	-0.099632	-0.101503	0.056483	-0.023027	-0.009826
Component 10	0.012536	0.049437	-0.093431	0.183938	0.182830	-0.006137	0.050009	-0.024924
Component 11	-0.209445	-0.235347	0.312530	-0.086994	-0.095515	-0.055173	-0.043641	0.006973
Component 12	0.388153	0.252654	0.069316	0.210952	0.208083	0.017225	0.060273	-0.040269
Component 13	0.149188	0.174006	-0.061219	0.011830	0.017057	0.021585	0.013671	0.151663
Component 14	0.223465	0.111619	0.522118	0.193132	0.178907	-0.002359	-0.063590	-0.018233
Component 15	0.101506	0.157026	-0.101766	0.004841	0.000206	-0.095466	-0.019765	0.014051
Component 16	0.135156	0.202739	-0.226999	0.005079	0.004138	0.000789	-0.167392	0.054119
Component 17	-0.069825	-0.145530	0.223401	0.030493	0.027700	-0.048285	-0.006666	0.130237

In [63]:

```
pca_scores = pca.transform(standard_data1_x)
pca_scores
```

Out[63]:
```
array([[-12.32059344,  -0.36954757,   0.37916276, ...,   0.94364479,
         -0.82492332,   1.17701995],
       [  7.34743079,  -7.50534251,  -4.93494667, ...,   1.07625205,
          0.34292626,  -0.81898672],
       [  7.5326384 ,   3.00641106,   3.21079888, ...,   0.32783211,
          0.69736066,   0.38245961],
       ...,
       [  6.48290237,  -6.20175209,  -0.55191897, ...,  -0.18453085,
          0.57748418,  -1.40641708],
       [  7.32215254,   5.98452075,  -2.93265794, ...,   0.47063844,
          0.16268049,  -1.48658746],
       [  8.65225668,  -6.39767012,  -0.25049863, ...,  -0.86539104,
          0.21171321,  -1.69000133]])
```

In [64]:
```
data_pc=pd.DataFrame(pca_scores)
```

In [65]:
```
data_pc
```

Out[65]:

	0	1	2	3	4	5	6	7	
0	-12.320593	-0.369548	0.379163	-0.537378	-4.611491	-2.850334	0.299161	0.976123	0.26
1	7.347431	-7.505343	-4.934947	-2.831809	-2.268518	2.033427	-2.831305	-1.495398	1.09
2	7.532638	3.006411	3.210799	0.265049	0.624286	-0.951773	-1.194467	0.174124	0.89
3	-4.803951	-3.779586	-2.289215	-0.793178	-2.268227	-1.432936	-2.419314	0.825711	-2.27
4	2.998904	-2.420024	-3.403583	7.941183	1.645077	-1.055211	-0.726078	-1.186998	0.52
...	...	...	...	...	...	...	...	...	
602	0.840366	7.139760	-3.100022	-1.075092	0.992142	0.360325	0.213444	0.956308	1.22
603	11.743582	-7.159023	0.547550	-3.280130	2.194944	-1.570850	0.452147	1.844019	1.63
604	6.482902	-6.201752	-0.551919	-3.684473	-0.762673	-0.209864	0.210706	-0.531676	0.95
605	7.322153	5.984521	-2.932658	-2.055989	0.152739	1.595523	0.873272	0.530457	2.02
606	8.652257	-6.397670	-0.250499	-2.420388	0.504726	-0.216966	-0.404685	0.921858	0.86

607 rows × 17 columns

In [66]:
```
data_pc.shape
```

Out[66]:
```
(607, 17)
```

Some more encoding

In [67]:
```
from category_encoders import TargetEncoder
encoder = TargetEncoder()
```

In [68]:
```
data_city=encoder.fit_transform(data['City'],data['Salary'])
data_cntryy=encoder.fit_transform(data['Cntry'],data['Salary'])
data_nat=encoder.fit_transform(data['Nat'],data['Salary'])
data_team=encoder.fit_transform(data['Team'],data['Salary'])
data_prst=encoder.fit_transform(data['Pr/St'],data['Salary'])
data_position=encoder.fit_transform(data['Position'],data['Salary'])
data['Hand']=data['Hand'].replace({'L':0,'R':1}).astype('int')
```

In [69]:
```
data['City']=data_city
data['Cntry']=data_cntryy
data['Nat']=data_nat
data['Team']=data_team
data['Pr/St']=data_prst
data['Position']=data_position
```

In [70]:
```
data.head()
```

Out[70]:

	Salary	Born	City	Pr/St	Cntry	Nat	Ht	Wt	DftYr
0	925000.0	97-01-30	2.206863e+06	2.338393e+06	2.253946e+06	2.226342e+06	74.0	190.0	2015.0
1	2250000.0	93-12-21	1.475042e+06	2.200632e+06	2.253946e+06	2.226342e+06	74.0	207.0	2012.0
2	7636250.0	88-04-16	4.149215e+06	3.113990e+06	1.987493e+06	2.009844e+06	72.0	218.0	2006.0
3	3500000.0	92-01-07	1.475042e+06	2.200632e+06	2.253946e+06	2.226342e+06	77.0	220.0	2010.0
4	1750000.0	94-03-29	1.873595e+06	2.200632e+06	2.253946e+06	2.226342e+06	76.0	217.0	2012.0

In [71]:
```
#sns.pairplot(data_pc,diag_kind="hist")
```

Feature Creation

In [72]:
```
ignore_feat = ['Salary']
data_categorical = list(set(list(data.select_dtypes(include=[object]).columns.valu
data_numerical = list(set(list(data.select_dtypes(include=[np.number]).columns.valu
```

In [73]:
```
data_categorical
```

Out[73]:
```
['Last Name', 'First Name', 'Born']
```

In [74]:
```
date_columns = [
    'Born']
# format dates
for col in date_columns:
    data.loc[:, col] = pd.to_datetime(data
                                      [col])
```

In [75]:
```
from datetime import date
def calculate_age(birthdate):
    today=date.today()
    age=today.year-birthdate.year-((today.month,today.day)<(birthdate.month,birthda
    return age
Age=data['Born'].apply(calculate_age)
Age
```

Out[75]:
```
0      25
1      28
2      34
3      30
4      28
       ..
607    29
608    37
609    35
610    39
611    29
Name: Born, Length: 607, dtype: int64
```

In [76]:
```
data['Born']=Age
```

Feature scaling

In [77]:
```
data_final=data.copy()
```

In [78]:
```
data_final.drop([
 'Last Name',

 'First Name',
 ],axis=1,inplace=True)
```

In [79]:
```
data_numerical = list(set(list(data.select_dtypes(include=[np.number]).columns.valu
```

In [80]:
```
scaler=StandardScaler()
scale = scaler.fit(data_final[data_numerical])
data_final[data_numerical] = scale.transform(data_final[data_numerical])
```

In [81]:
```
data_final.head()
```

Out[81]:

	Salary	Born	City	Pr/St	Cntry	Nat	Ht	Wt	DftY
0	925000.0	-1.473546	-0.144318	0.238537	0.166804	0.042088	0.521519	-0.721232	1.51869
1	2250000.0	-0.798495	-0.986221	-0.047152	0.166804	0.042088	0.521519	0.429238	0.75504
2	7636250.0	0.551607	2.090207	1.846980	-0.938496	-0.966695	-0.493103	1.173660	-0.77224
3	3500000.0	-0.348461	-0.986221	-0.047152	0.166804	0.042088	2.043452	1.309009	0.24595
4	1750000.0	-0.798495	-0.527717	-0.047152	0.166804	0.042088	1.536141	1.105985	0.75504

Feature Selection

In [82]:
```
X = data_final.drop(['Salary'], axis=1)
y = data_final['Salary']
```

In [83]:
```
X.shape, y.shape
```

Out[83]:
```
((607, 151), (607,))
```

In [84]:
```
# importing the models
```

```
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
from sklearn.linear_model import LinearRegression
```

In [85]:
```
# calling the linear regression model

lreg = LinearRegression()
```

Forward Selection

In [86]:
```
sfs1 = sfs(lreg, k_features=25, forward=True, verbose=2, scoring='neg_mean_squared_
```

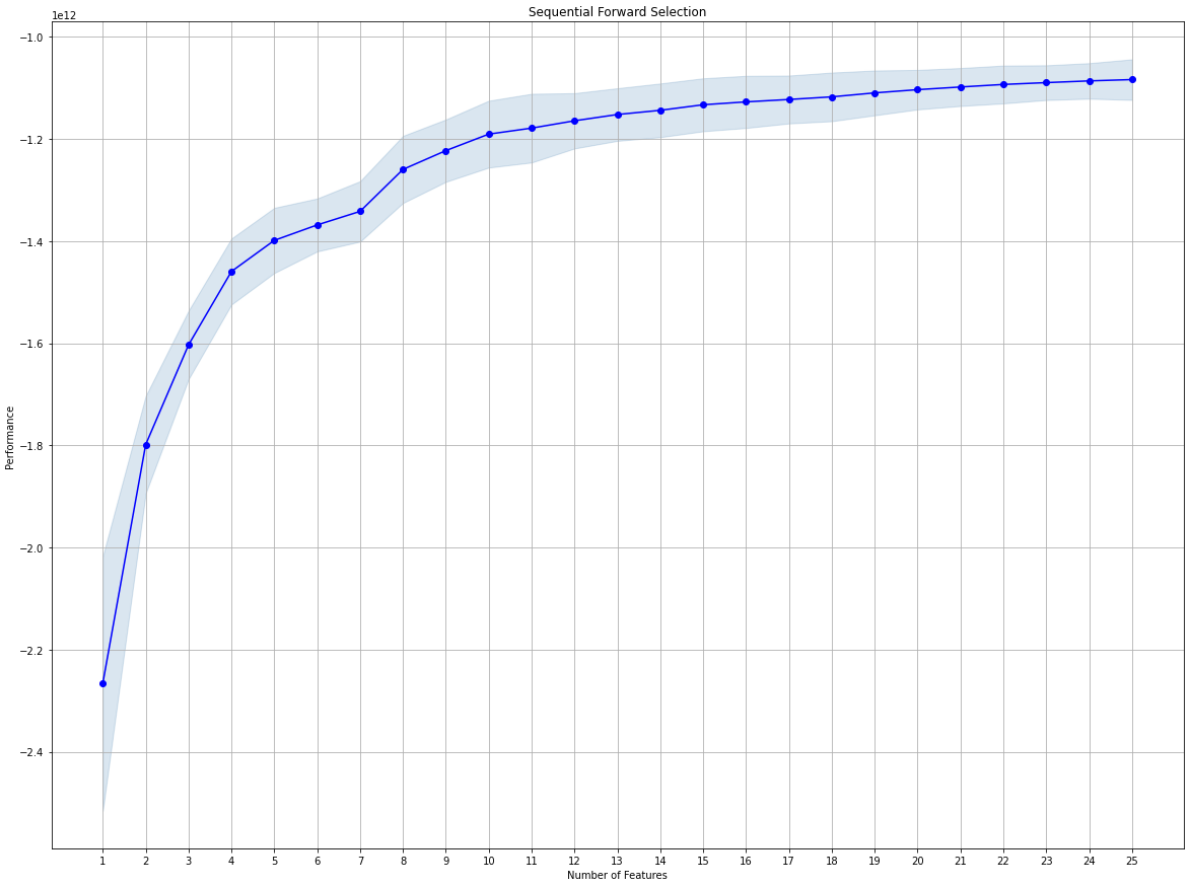
In [87]:
```
sfs1 = sfs1.fit(X, y)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 151 out of 151 | elapsed:    0.9s finished

[2022-07-12 02:44:46] Features: 1/25 -- score: -2265985927616.8213[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 150 out of 150 | elapsed:    0.8s finished

[2022-07-12 02:44:47] Features: 2/25 -- score: -1798479108514.698[Parallel(n_jobs=
1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 149 out of 149 | elapsed:    0.8s finished

[2022-07-12 02:44:48] Features: 3/25 -- score: -1603067302726.8203[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 148 out of 148 | elapsed:    0.8s finished

[2022-07-12 02:44:49] Features: 4/25 -- score: -1459423233538.688[Parallel(n_jobs=
1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 147 out of 147 | elapsed:    0.8s finished

[2022-07-12 02:44:50] Features: 5/25 -- score: -1398540100908.4219[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 146 out of 146 | elapsed:    0.8s finished

[2022-07-12 02:44:51] Features: 6/25 -- score: -1368294485331.1018[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 145 out of 145 | elapsed:    0.8s finished

[2022-07-12 02:44:52] Features: 7/25 -- score: -1341740151229.9226[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 144 out of 144 | elapsed:    0.8s finished

[2022-07-12 02:44:53] Features: 8/25 -- score: -1259560365279.5337[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 143 out of 143 | elapsed:    0.8s finished

[2022-07-12 02:44:54] Features: 9/25 -- score: -1222762137862.141[Parallel(n_jobs=
1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 142 out of 142 | elapsed:    0.9s finished

[2022-07-12 02:44:55] Features: 10/25 -- score: -1190466740981.2134[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 141 out of 141 | elapsed:    0.8s finished

[2022-07-12 02:44:55] Features: 11/25 -- score: -1178703175391.6978[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    0.8s finished

[2022-07-12 02:44:56] Features: 12/25 -- score: -1164355394136.6934[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 139 out of 139 | elapsed:    0.8s finished
```

```
[2022-07-12 02:44:57] Features: 13/25 -- score: -1152080944820.8364[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 138 out of 138 | elapsed:    0.9s finished

[2022-07-12 02:44:58] Features: 14/25 -- score: -1143824107194.918[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 137 out of 137 | elapsed:    0.8s finished

[2022-07-12 02:44:59] Features: 15/25 -- score: -1132961024183.6794[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 136 out of 136 | elapsed:    0.8s finished

[2022-07-12 02:45:00] Features: 16/25 -- score: -1127317424517.1704[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 135 out of 135 | elapsed:    0.9s finished

[2022-07-12 02:45:01] Features: 17/25 -- score: -1122625421257.7642[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 134 out of 134 | elapsed:    0.9s finished

[2022-07-12 02:45:02] Features: 18/25 -- score: -1117480118508.775[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 133 out of 133 | elapsed:    0.9s finished

[2022-07-12 02:45:03] Features: 19/25 -- score: -1109702380780.8833[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 132 out of 132 | elapsed:    0.9s finished

[2022-07-12 02:45:04] Features: 20/25 -- score: -1103391327969.7976[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 131 out of 131 | elapsed:    0.9s finished

[2022-07-12 02:45:05] Features: 21/25 -- score: -1098175719749.6484[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 130 out of 130 | elapsed:    0.9s finished

[2022-07-12 02:45:06] Features: 22/25 -- score: -1093195564891.5153[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 129 out of 129 | elapsed:    0.9s finished

[2022-07-12 02:45:07] Features: 23/25 -- score: -1089622456489.9625[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 128 out of 128 | elapsed:    0.9s finished

[2022-07-12 02:45:08] Features: 24/25 -- score: -1086243840835.066[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 127 out of 127 | elapsed:    0.9s finished

[2022-07-12 02:45:09] Features: 25/25 -- score: -1083656341457.1848
```

In [88]:
```
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
```

```
fig1 = plot_sfs(sfs1.get_metric_dict(confidence_interval=0.95), kind='std_err',figs

plt.title('Sequential Forward Selection')
plt.grid()

plt.show()
```



In [89]:
```
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

In [90]:
```
df = pd.DataFrame.from_dict(sfs1.get_metric_dict()).T
df[["feature_idx","feature_names","avg_score"]]
```

Out[90]:

	feature_idx	feature_names	avg_score
1	(122,)	(xGF,)	-2265985927616.821289
2	(7, 122)	(DftYr, xGF)	-1798479108514.697998
3	(1, 7, 122)	(City, DftYr, xGF)	-1603067302726.820312
4	(1, 7, 13, 122)	(City, DftYr, GP, xGF)	-1459423233538.687988
5	(1, 7, 9, 13, 122)	(City, DftYr, Ovrl, GP, xGF)	-1398540100908.421875
6	(1, 7, 9, 12, 13, 122)	(City, DftYr, Ovrl, Team, GP, xGF)	-1368294485331.101807
7	(1, 7, 9, 12, 13, 122, 134)	(City, DftYr, Ovrl, Team, GP, xGF, FOW)	-1341740151229.922607
8	(1, 7, 9, 12, 13, 44, 122, 134)	(City, DftYr, Ovrl, Team, GP, iSCF, xGF, FOW)	-1259560365279.533691
9	(1, 7, 9, 12, 13, 25, 44, 122, 134)	(City, DftYr, Ovrl, Team, GP, TOI/GP, iSCF, xGF, FOW)	-1222762137862.141113
10	(1, 7, 9, 12, 13, 25, 44, 54, 122, 134)	(City, DftYr, Ovrl, Team, GP, TOI/GP, iSCF, iHDf, xGF, FOW)	-1190466740981.213379
11	(1, 7, 9, 12, 13, 25, 44, 54, 60, 122, 134)	(City, DftYr, Ovrl, Team, GP, TOI/GP, iSCF, iHDf, iTKA.1, xGF, FOW)	-1178703175391.697754
12	(1, 7, 9, 12, 13, 25, 44, 54, 60, 122, 134, 135)	(City, DftYr, Ovrl, Team, GP, TOI/GP, iSCF, iHDf, iTKA.1, xGF, FOW, FOL)	-1164355394136.693359
13	(1, 7, 9, 12, 13, 25, 44, 54, 60, 119, 122, 134, 135)	(City, DftYr, Ovrl, Team, GP, TOI/GP, iSCF, iHDf, iTKA.1, FA, xGF, FOW, FOL)	-1152080944820.836426
14	(1, 2, 7, 9, 12, 13, 25, 44, 54, 60, 119, 122, 134, 135)	(City, Pr/St, DftYr, Ovrl, Team, GP, TOI/GP, iSCF, iHDf, iTKA.1, FA, xGF, FOW, FOL)	-1143824107194.917969
15	(1, 2, 7, 9, 12, 13, 23, 25, 44, 54, 60, 119, 122, 134, 135)	(City, Pr/St, DftYr, Ovrl, Team, GP, TOI, TOI/GP, iSCF, iHDf, iTKA.1, FA, xGF, FOW, FOL)	-1132961024183.679443
16	(1, 2, 7, 9, 12, 13, 23, 25, 44, 54, 60, 62, 119, 122, 134, 135)	(City, Pr/St, DftYr, Ovrl, Team, GP, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, FA, xGF, FOW, FOL)	-1127317424517.17041
17	(1, 2, 7, 9, 12, 13, 23, 25, 44, 54, 60, 62, 119, 122, 134, 135, 141)	(City, Pr/St, DftYr, Ovrl, Team, GP, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, FA, xGF, FOW, FOL, PEND)	-1122625421257.76416
18	(1, 2, 7, 9, 12, 13, 23, 25, 44, 54, 60, 62, 90, 119, 122, 134, 135, 141)	(City, Pr/St, DftYr, Ovrl, Team, GP, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, G.Snap, FA, xGF, FOW, FOL, PEND)	-1117480118508.774902
19	(1, 2, 7, 9, 12, 13, 23, 25, 44, 54, 60, 62, 90, 119, 122, 134, 135, 141, 142)	(City, Pr/St, DftYr, Ovrl, Team, GP, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, G.Snap, FA, xGF, FOW, FOL, PEND, OPS)	-1109702380780.883301
20	(1, 2, 7, 9, 12, 13, 23, 25, 44, 54, 60, 62, 83, 90, 119, 122, 134, 135, 141, 142)	(City, Pr/St, DftYr, Ovrl, Team, GP, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, GWG, G.Snap, FA, xGF, FOW, FOL, PEND, OPS)	-1103391327969.797607
21	(1, 2, 7, 9, 12, 13, 22, 23, 25, 44, 54, 60, 62, 83, 90, 119, 122, 134, 135, 141, 142)	(City, Pr/St, DftYr, Ovrl, Team, GP, Shifts, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, GWG, G.Snap, FA, xGF, FOW, FOL, PEND, OPS)	-1098175719749.648438
22	(1, 2, 7, 9, 12, 13, 22, 23, 25, 44, 54, 60, 62, 83, 90, 98, 119, 122, 134, 135, 141, 142)	(City, Pr/St, DftYr, Ovrl, Team, GP, Shifts, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, GWG, G.Snap, S.Bkhd, FA, xGF, FOW, FOL, PEND, OPS)	-1093195564891.515259

	feature_idx	feature_names	avg_score
23	(1, 2, 7, 9, 12, 13, 22, 23, 25, 44, 54, 60, 62, 82, 83, 90, 98, 119, 122, 134, 135, 141, 142)	(City, Pr/St, DftYr, Ovrl, Team, GP, Shifts, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, 1G, GWG, G.Snap, S.Bkhd, FA, xGF, FOW, FOL, PEND, OPS)	-1089622456489.962524
24	(1, 2, 7, 9, 12, 13, 22, 23, 25, 44, 54, 60, 62, 82, 83, 90, 98, 106, 119, 122, 134, 135, 141, 142)	(City, Pr/St, DftYr, Ovrl, Team, GP, Shifts, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, 1G, GWG, G.Snap, S.Bkhd, iPenD, FA, xGF, FOW, FOL, PEND, OPS)	-1086243840835.06604
25	(1, 2, 6, 7, 9, 12, 13, 22, 23, 25, 44, 54, 60, 62, 82, 83, 90, 98, 106, 119, 122, 134, 135, 141, 142)	(City, Pr/St, Wt, DftYr, Ovrl, Team, GP, Shifts, TOI, TOI/GP, iSCF, iHDf, iTKA.1, BLK%, 1G, GWG, G.Snap, S.Bkhd, iPenD, FA, xGF, FOW, FOL, PEND, OPS)	-1083656341457.184814

In [91]:
```
# Lets access the indices of the best features directly via the k_feature_idx_ attr
sfs1.k_feature_names_, sfs1.k_feature_idx_
```

Out[91]:
```
(('City',
  'Pr/St',
  'Wt',
  'DftYr',
  'Ovrl',
  'Team',
  'GP',
  'Shifts',
  'TOI',
  'TOI/GP',
  'iSCF',
  'iHDf',
  'iTKA.1',
  'BLK%',
  '1G',
  'GWG',
  'G.Snap',
  'S.Bkhd',
  'iPenD',
  'FA',
  'xGF',
  'FOW',
  'FOL',
  'PEND',
  'OPS'),
 (1,
  2,
  6,
  7,
  9,
  12,
  13,
  22,
  23,
  25,
  44,
  54,
  60,
  62,
  82,
  83,
  90,
  98,
  106,
  119,
  122,
  134,
  135,
  141,
  142))
```

In [92]:
```
selected_feat= X.columns[list(sfs1.k_feature_idx_)]

print(selected_feat)
print("----------------------------------------------------")
print("Total Number of Selected Features: {}".format(len(selected_feat)))
```

```
Index(['City', 'Pr/St', 'Wt', 'DftYr', 'Ovrl', 'Team', 'GP', 'Shifts', 'TOI', 'TO
I/GP', 'iSCF', 'iHDf', 'iTKA.1', 'BLK%', '1G', 'GWG', 'G.Snap', 'S.Bkhd', 'iPenD',
'FA', 'xGF', 'FOW', 'FOL', 'PEND', 'OPS'], dtype='object')
----------------------------------------------------
Total Number of Selected Features: 25
```

Backward Selection

In [93]:
```
sfs2 = sfs(lreg, k_features=25, forward=False, verbose=2, scoring='neg_mean_squared
```

In [94]:
```
sfs2 = sfs2.fit(X, y)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 151 out of 151 | elapsed:    3.8s finished

[2022-07-12 02:45:13] Features: 150/25 -- score: -1691142452220.444[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 150 out of 150 | elapsed:    3.7s finished

[2022-07-12 02:45:17] Features: 149/25 -- score: -1647343639368.702[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 149 out of 149 | elapsed:    3.7s finished

[2022-07-12 02:45:21] Features: 148/25 -- score: -1613106032492.286[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 148 out of 148 | elapsed:    3.6s finished

[2022-07-12 02:45:25] Features: 147/25 -- score: -1582587497743.73[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 147 out of 147 | elapsed:    3.5s finished

[2022-07-12 02:45:28] Features: 146/25 -- score: -1554940895715.7812[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 146 out of 146 | elapsed:    3.5s finished

[2022-07-12 02:45:32] Features: 145/25 -- score: -1532999316566.421[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 145 out of 145 | elapsed:    3.7s finished

[2022-07-12 02:45:36] Features: 144/25 -- score: -1509688017724.8062[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 144 out of 144 | elapsed:    3.6s finished

[2022-07-12 02:45:40] Features: 143/25 -- score: -1490529413605.1848[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 143 out of 143 | elapsed:    3.6s finished

[2022-07-12 02:45:43] Features: 142/25 -- score: -1475571804307.8535[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 142 out of 142 | elapsed:    3.6s finished

[2022-07-12 02:45:47] Features: 141/25 -- score: -1461854182263.6755[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 141 out of 141 | elapsed:    3.6s finished

[2022-07-12 02:45:51] Features: 140/25 -- score: -1448046364530.0652[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    3.5s finished

[2022-07-12 02:45:54] Features: 139/25 -- score: -1432989477967.8306[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 139 out of 139 | elapsed:    3.4s finished
```

```
[2022-07-12 02:45:58] Features: 138/25 -- score: -1419390463413.224[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 138 out of 138 | elapsed:    3.4s finished

[2022-07-12 02:46:01] Features: 137/25 -- score: -1409350436762.9392[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 137 out of 137 | elapsed:    3.5s finished

[2022-07-12 02:46:05] Features: 136/25 -- score: -1400781220390.5078[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 136 out of 136 | elapsed:    3.3s finished

[2022-07-12 02:46:08] Features: 135/25 -- score: -1391826413880.684[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 135 out of 135 | elapsed:    3.7s finished

[2022-07-12 02:46:12] Features: 134/25 -- score: -1383261232014.6538[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 134 out of 134 | elapsed:    3.3s finished

[2022-07-12 02:46:16] Features: 133/25 -- score: -1376048607771.8074[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 133 out of 133 | elapsed:    3.2s finished

[2022-07-12 02:46:19] Features: 132/25 -- score: -1369249622612.551[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 132 out of 132 | elapsed:    3.1s finished

[2022-07-12 02:46:22] Features: 131/25 -- score: -1362934949461.5454[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 131 out of 131 | elapsed:    3.1s finished

[2022-07-12 02:46:25] Features: 130/25 -- score: -1356595193270.5188[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 130 out of 130 | elapsed:    3.1s finished

[2022-07-12 02:46:29] Features: 129/25 -- score: -1350166079716.2542[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 129 out of 129 | elapsed:    3.0s finished

[2022-07-12 02:46:32] Features: 128/25 -- score: -1344370424999.3782[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 128 out of 128 | elapsed:    2.9s finished

[2022-07-12 02:46:35] Features: 127/25 -- score: -1338815583393.5479[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 127 out of 127 | elapsed:    2.9s finished

[2022-07-12 02:46:38] Features: 126/25 -- score: -1333004527678.0845[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 126 out of 126 | elapsed:    2.8s finished
```

```
[2022-07-12 02:46:40] Features: 125/25 -- score: -1327246110320.2603[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 125 out of 125 | elapsed:    2.7s finished

[2022-07-12 02:46:43] Features: 124/25 -- score: -1320882542577.3823[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 124 out of 124 | elapsed:    2.7s finished

[2022-07-12 02:46:46] Features: 123/25 -- score: -1313822808100.114[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 123 out of 123 | elapsed:    2.6s finished

[2022-07-12 02:46:49] Features: 122/25 -- score: -1307016991802.317[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 122 out of 122 | elapsed:    2.6s finished

[2022-07-12 02:46:52] Features: 121/25 -- score: -1305948937092.5083[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 121 out of 121 | elapsed:    2.5s finished

[2022-07-12 02:46:54] Features: 120/25 -- score: -1298495988815.464[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 120 out of 120 | elapsed:    2.5s finished

[2022-07-12 02:46:57] Features: 119/25 -- score: -1292076448981.8022[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 119 out of 119 | elapsed:    2.5s finished

[2022-07-12 02:46:59] Features: 118/25 -- score: -1286148639324.57[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 118 out of 118 | elapsed:    2.6s finished

[2022-07-12 02:47:02] Features: 117/25 -- score: -1275403929823.8403[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 117 out of 117 | elapsed:    2.4s finished

[2022-07-12 02:47:05] Features: 116/25 -- score: -1264769155403.746[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 116 out of 116 | elapsed:    2.4s finished

[2022-07-12 02:47:07] Features: 115/25 -- score: -1265715238849.425[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 115 out of 115 | elapsed:    2.3s finished

[2022-07-12 02:47:10] Features: 114/25 -- score: -1263899545230.9758[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 114 out of 114 | elapsed:    2.3s finished

[2022-07-12 02:47:12] Features: 113/25 -- score: -1263126310541.1208[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
```

```
[Parallel(n_jobs=1)]: Done 113 out of 113 | elapsed:    2.2s finished

[2022-07-12 02:47:14] Features: 112/25 -- score: -1265866867966.144[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 112 out of 112 | elapsed:    2.2s finished

[2022-07-12 02:47:17] Features: 111/25 -- score: -1266263018690.2817[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 111 out of 111 | elapsed:    2.2s finished

[2022-07-12 02:47:19] Features: 110/25 -- score: -1268807073738.3115[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:    2.1s finished

[2022-07-12 02:47:21] Features: 109/25 -- score: -1270387703112.8335[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 109 out of 109 | elapsed:    2.1s finished

[2022-07-12 02:47:23] Features: 108/25 -- score: -1271520580887.7944[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 108 out of 108 | elapsed:    2.1s finished

[2022-07-12 02:47:26] Features: 107/25 -- score: -1279357274413.4807[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 107 out of 107 | elapsed:    2.0s finished

[2022-07-12 02:47:28] Features: 106/25 -- score: -1275061722657.7358[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 106 out of 106 | elapsed:    2.0s finished

[2022-07-12 02:47:30] Features: 105/25 -- score: -1274183929543.1577[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 105 out of 105 | elapsed:    2.0s finished

[2022-07-12 02:47:32] Features: 104/25 -- score: -1249311717191.823[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 104 out of 104 | elapsed:    1.9s finished

[2022-07-12 02:47:34] Features: 103/25 -- score: -1256781280252.4285[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 103 out of 103 | elapsed:    1.9s finished

[2022-07-12 02:47:36] Features: 102/25 -- score: -1242037473257.542[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 102 out of 102 | elapsed:    1.9s finished

[2022-07-12 02:47:38] Features: 101/25 -- score: -1258133286482.3057[Parallel(n_jo
bs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 101 out of 101 | elapsed:    1.8s finished

[2022-07-12 02:47:40] Features: 100/25 -- score: -1249681437425.579[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    1.8s finished

[2022-07-12 02:47:42] Features: 99/25 -- score: -1242482385270.748[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  99 out of  99 | elapsed:    1.8s finished

[2022-07-12 02:47:43] Features: 98/25 -- score: -1236732466111.7734[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  98 out of  98 | elapsed:    1.7s finished

[2022-07-12 02:47:45] Features: 97/25 -- score: -1233166871990.907[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  97 out of  97 | elapsed:    1.7s finished

[2022-07-12 02:47:47] Features: 96/25 -- score: -1236022410395.2607[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  96 out of  96 | elapsed:    1.7s finished

[2022-07-12 02:47:49] Features: 95/25 -- score: -1232931752793.2397[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  95 out of  95 | elapsed:    1.6s finished

[2022-07-12 02:47:51] Features: 94/25 -- score: -1228711743175.2058[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  94 out of  94 | elapsed:    1.6s finished

[2022-07-12 02:47:52] Features: 93/25 -- score: -1223230565630.9658[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  93 out of  93 | elapsed:    1.6s finished

[2022-07-12 02:47:54] Features: 92/25 -- score: -1219231389301.5073[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  92 out of  92 | elapsed:    1.6s finished

[2022-07-12 02:47:56] Features: 91/25 -- score: -1201039653965.9792[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  91 out of  91 | elapsed:    1.5s finished

[2022-07-12 02:47:57] Features: 90/25 -- score: -1210043670906.2827[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed:    1.4s finished

[2022-07-12 02:47:59] Features: 89/25 -- score: -1205288470320.5562[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  89 out of  89 | elapsed:    1.4s finished

[2022-07-12 02:48:00] Features: 88/25 -- score: -1211324135517.0537[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  88 out of  88 | elapsed:    1.4s finished

[2022-07-12 02:48:02] Features: 87/25 -- score: -1198399485104.2[Parallel(n_jobs=
```

```
1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  87 out of  87 | elapsed:    1.3s finished

[2022-07-12 02:48:03] Features: 86/25 -- score: -1206868779542.2148[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  86 out of  86 | elapsed:    1.3s finished

[2022-07-12 02:48:05] Features: 85/25 -- score: -1200364091154.6763[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  85 out of  85 | elapsed:    1.3s finished

[2022-07-12 02:48:06] Features: 84/25 -- score: -1189659337571.8274[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  84 out of  84 | elapsed:    1.3s finished

[2022-07-12 02:48:07] Features: 83/25 -- score: -1192090159379.6382[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  83 out of  83 | elapsed:    1.3s finished

[2022-07-12 02:48:09] Features: 82/25 -- score: -1191741645778.4014[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  82 out of  82 | elapsed:    1.6s finished

[2022-07-12 02:48:10] Features: 81/25 -- score: -1189837980888.0674[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  81 out of  81 | elapsed:    1.3s finished

[2022-07-12 02:48:12] Features: 80/25 -- score: -1184181950642.8745[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    1.2s finished

[2022-07-12 02:48:13] Features: 79/25 -- score: -1173177892746.5881[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  79 out of  79 | elapsed:    1.1s finished

[2022-07-12 02:48:14] Features: 78/25 -- score: -1168727855649.4707[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  78 out of  78 | elapsed:    1.1s finished

[2022-07-12 02:48:16] Features: 77/25 -- score: -1168174354689.4749[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  77 out of  77 | elapsed:    1.0s finished

[2022-07-12 02:48:17] Features: 76/25 -- score: -1159939287411.3386[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  76 out of  76 | elapsed:    1.0s finished

[2022-07-12 02:48:18] Features: 75/25 -- score: -1164228940330.189[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  75 out of  75 | elapsed:    0.9s finished
```

```
[2022-07-12 02:48:19] Features: 74/25 -- score: -1158181277563.8306[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  74 out of  74 | elapsed:    0.9s finished

[2022-07-12 02:48:20] Features: 73/25 -- score: -1153710820584.9204[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  73 out of  73 | elapsed:    0.9s finished

[2022-07-12 02:48:21] Features: 72/25 -- score: -1150021581421.573[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  72 out of  72 | elapsed:    0.9s finished

[2022-07-12 02:48:22] Features: 71/25 -- score: -1147123725485.7637[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  71 out of  71 | elapsed:    0.9s finished

[2022-07-12 02:48:23] Features: 70/25 -- score: -1145802495531.2686[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  70 out of  70 | elapsed:    0.8s finished

[2022-07-12 02:48:24] Features: 69/25 -- score: -1142529549221.1948[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  69 out of  69 | elapsed:    0.8s finished

[2022-07-12 02:48:25] Features: 68/25 -- score: -1139525606642.1602[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  68 out of  68 | elapsed:    0.8s finished

[2022-07-12 02:48:26] Features: 67/25 -- score: -1136849824216.029[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  67 out of  67 | elapsed:    0.8s finished

[2022-07-12 02:48:27] Features: 66/25 -- score: -1134264114709.8755[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  66 out of  66 | elapsed:    0.8s finished

[2022-07-12 02:48:27] Features: 65/25 -- score: -1131594635499.2466[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  65 out of  65 | elapsed:    0.7s finished

[2022-07-12 02:48:28] Features: 64/25 -- score: -1129511411063.1062[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  64 out of  64 | elapsed:    0.7s finished

[2022-07-12 02:48:29] Features: 63/25 -- score: -1128111135710.9785[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  63 out of  63 | elapsed:    0.7s finished

[2022-07-12 02:48:30] Features: 62/25 -- score: -1126502223901.58[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  62 out of  62 | elapsed:    0.6s finished
```

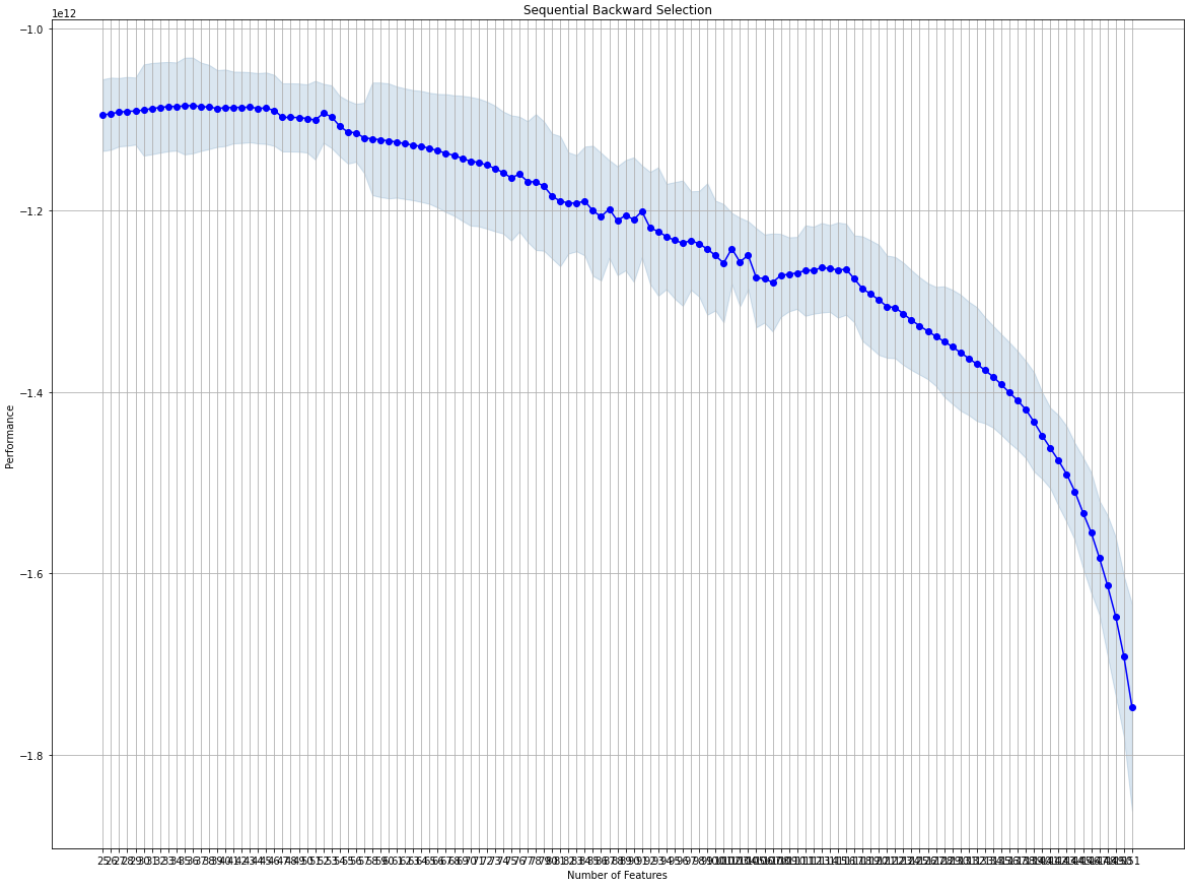
```
[2022-07-12 02:48:31] Features: 61/25 -- score: -1124705704692.721[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  61 out of  61 | elapsed:    0.6s finished

[2022-07-12 02:48:31] Features: 60/25 -- score: -1123409360387.043[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:    0.6s finished

[2022-07-12 02:48:32] Features: 59/25 -- score: -1122224520551.6685[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  59 out of  59 | elapsed:    0.6s finished

[2022-07-12 02:48:33] Features: 58/25 -- score: -1121125966322.2656[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  58 out of  58 | elapsed:    0.6s finished

[2022-07-12 02:48:33] Features: 57/25 -- score: -1120099939111.1646[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  57 out of  57 | elapsed:    0.6s finished

[2022-07-12 02:48:34] Features: 56/25 -- score: -1114508388405.9683[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  56 out of  56 | elapsed:    0.6s finished

[2022-07-12 02:48:35] Features: 55/25 -- score: -1113632529013.978[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  55 out of  55 | elapsed:    0.5s finished

[2022-07-12 02:48:35] Features: 54/25 -- score: -1107151833031.629[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed:    0.5s finished

[2022-07-12 02:48:36] Features: 53/25 -- score: -1096721682851.801[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  53 out of  53 | elapsed:    0.5s finished

[2022-07-12 02:48:37] Features: 52/25 -- score: -1093027734796.9844[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  52 out of  52 | elapsed:    0.5s finished

[2022-07-12 02:48:37] Features: 51/25 -- score: -1100562693912.2788[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  51 out of  51 | elapsed:    0.5s finished

[2022-07-12 02:48:38] Features: 50/25 -- score: -1098681793758.9861[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:    0.4s finished

[2022-07-12 02:48:38] Features: 49/25 -- score: -1097717429009.4899[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
```

```
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed:    0.4s finished

[2022-07-12 02:48:39] Features: 48/25 -- score: -1097484130671.9896[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    0.4s finished

[2022-07-12 02:48:39] Features: 47/25 -- score: -1097484130671.99[Parallel(n_jobs=
1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed:    0.4s finished

[2022-07-12 02:48:40] Features: 46/25 -- score: -1089795797200.0491[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed:    0.4s finished

[2022-07-12 02:48:40] Features: 45/25 -- score: -1087357899730.0328[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:    0.4s finished

[2022-07-12 02:48:41] Features: 44/25 -- score: -1087444478391.8372[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed:    0.4s finished

[2022-07-12 02:48:41] Features: 43/25 -- score: -1086318873180.5469[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed:    0.3s finished

[2022-07-12 02:48:42] Features: 42/25 -- score: -1086457605926.2533[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:    0.3s finished

[2022-07-12 02:48:42] Features: 41/25 -- score: -1086596703947.5895[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed:    0.3s finished

[2022-07-12 02:48:43] Features: 40/25 -- score: -1086859734911.9629[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    0.3s finished

[2022-07-12 02:48:43] Features: 39/25 -- score: -1087719297359.9229[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed:    0.3s finished

[2022-07-12 02:48:43] Features: 38/25 -- score: -1085994038016.4164[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed:    0.3s finished

[2022-07-12 02:48:44] Features: 37/25 -- score: -1085772314822.2234[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed:    0.3s finished

[2022-07-12 02:48:44] Features: 36/25 -- score: -1084276640594.9625[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:    0.2s finished

[2022-07-12 02:48:45] Features: 35/25 -- score: -1084979105353.4388[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed:    0.3s finished

[2022-07-12 02:48:45] Features: 34/25 -- score: -1085486054065.778[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed:    0.2s finished

[2022-07-12 02:48:45] Features: 33/25 -- score: -1085609416971.3284[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed:    0.2s finished

[2022-07-12 02:48:46] Features: 32/25 -- score: -1086632398434.6302[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:    0.2s finished

[2022-07-12 02:48:46] Features: 31/25 -- score: -1087822114752.251[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed:    0.2s finished

[2022-07-12 02:48:46] Features: 30/25 -- score: -1089383062841.3308[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.3s finished

[2022-07-12 02:48:47] Features: 29/25 -- score: -1090559905132.8815[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed:    0.3s finished

[2022-07-12 02:48:47] Features: 28/25 -- score: -1090851822548.7175[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:    0.3s finished

[2022-07-12 02:48:47] Features: 27/25 -- score: -1091751207867.9808[Parallel(n_job
s=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    0.3s finished

[2022-07-12 02:48:48] Features: 26/25 -- score: -1093389616577.717[Parallel(n_jobs
=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:    0.2s finished

[2022-07-12 02:48:48] Features: 25/25 -- score: -1094957082864.4557
```

In [95]:

```
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs

fig1 = plot_sfs(sfs2.get_metric_dict(confidence_interval=0.95), kind='std_err',figs

plt.title('Sequential Backward Selection')
plt.grid()

plt.show()
```



In [96]:
```
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
```

In [97]:
```
df = pd.DataFrame.from_dict(sfs2.get_metric_dict()).T
df[["feature_idx","feature_names","avg_score"]]
```

Out[97]:

	feature_idx	feature_names	avg_score
151	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, G, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iRS, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iHA, iHDf, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOW.1, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Over, Wide, S.Bkhd, S.Dflct, ...)	-1747319834967.112793
150	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, G, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iRS, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iHA, iHDf, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOW.1, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, ...)	-1691142452220.444092
149	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, G, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iRS, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iHDf, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOW.1, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, ...)	-1647343639368.701904

	feature_idx	feature_names	avg_score
148	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, G, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iRS, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iHDf, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOW.1, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, ...)	-1613106032492.285889
147	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, 102, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, G, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iHDf, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOW.1, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Tip, ...)	-1582587497743.72998
146	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, 103, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, G, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iHDf, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOW.1, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrap, ...)	-1554940895715.78125

	feature_idx	feature_names	avg_score
145	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, 103, 104, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, G, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iHDf, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrap, S.Wrst, ...)	-1532999316566.420898
144	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, 103, 104, 105, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iHDf, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrap, S.Wrst, iPenT, ...)	-1509688017724.806152
143	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, 103, 104, 105, 106, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrap, S.Wrst, iPenT, iPenD, ...)	-1490529413605.184814

	feature_idx	feature_names	avg_score
142	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 97, 98, 99, 100, 101, 103, 104, 105, 106, 107, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Tip, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrap, S.Wrst, iPenT, iPenD, iPENT, ...)	-1475571804307.853516
141	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 103, 104, 105, 106, 107, 108, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrap, S.Wrst, iPenT, iPenD, iPENT, iPEND, ...)	-1461854182263.675537
140	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 104, 105, 106, 107, 108, 109, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPENT, iPEND, iPenDf, ...)	-1448046364530.065186

	feature_idx	feature_names	avg_score
139	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 104, 105, 106, 107, 108, 110, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPENT, iPEND, NPD, ...)	-1432989477967.830566
138	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 104, 105, 106, 107, 108, 110, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOW, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPENT, iPEND, NPD, ...)	-1419390463413.224121
137	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 104, 105, 106, 107, 108, 110, 111, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPENT, iPEND, NPD, Min, ...)	-1409350436762.939209

	feature_idx	feature_names	avg_score
136	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 104, 105, 106, 108, 110, 111, 112, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, ...)	-1400781220390.507812
135	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 104, 105, 106, 108, 110, 111, 112, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, ...)	-1391826413880.684082
134	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 104, 105, 106, 108, 110, 111, 112, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, ...)	-1383261232014.653809

	feature_idx	feature_names	avg_score
133	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 99, 100, 101, 104, 105, 106, 108, 110, 111, 112, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Dflct, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, ...)	-1376048607771.807373
132	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 100, 101, 104, 105, 106, 108, 110, 111, 112, 113, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, Shifts, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, ...)	-1369249622612.551025
131	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 100, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, iFOL.1, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, ...)	-1362934949461.54541

	feature_idx	feature_names	avg_score
130	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 97, 98, 100, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Bkhd, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, ...)	-1356595193270.518799
129	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 70, 71, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 100, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, ...)	-1350166079716.25415
128	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 100, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Slap, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, ...)	-1344370424999.378174

	feature_idx	feature_names	avg_score
127	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, ...)	-1338815583393.547852
126	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, ...)	-1333004527678.084473
125	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iGVA.1, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, ...)	-1327246110320.260254
124	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, ...)	-1320882542577.382324

	feature_idx	feature_names	avg_score
123	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55, 56, 57, 58, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iMiss, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, ...)	-1313822808100.114014
122	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, 123, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, xGA, ...)	-1307016991802.316895
121	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, 123, 125, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOI, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, xGA, SCA, ...)	-1305948937092.508301
120	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 67, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, 123, 125, 127, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, FO%, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, xGA, SCA, GA, ...)	-1298495988815.464111

	feature_idx	feature_names	avg_score
119	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, 123, 125, 127, 128, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, PIM, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, xGA, SCA, GA, RBF, ...)	-1292076448981.802246
118	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 97, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, 123, 125, 127, 128, 129, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, Wide, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, xGA, SCA, GA, RBF, RBA, ...)	-1286148639324.570068
117	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, 123, 125, 127, 128, 129, 130, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, xGA, SCA, GA, RBF, RBA, RSF, ...)	-1275403929823.840332

	feature_idx	feature_names	avg_score
116	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, OTG, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, ...)	-1264769155403.746094
115	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 115, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, Game, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, ...)	-1265715238849.425049
114	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 92, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, ...)	-1263899545230.97583

	feature_idx	feature_names	avg_score
113	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 114, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 134, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, Misc, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, FOW, ...)	-1263126310541.12085
112	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 113, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 134, 137, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, Match, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, FOW, HA, ...)	-1265866867966.144043
111	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 134, 137, 138, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, FOW, HA, GVA, ...)	-1266263018690.281738

	feature_idx	feature_names	avg_score
110	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, PTS, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, ...)	-1268807073738.311523
109	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 94, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, CBar , Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, ...)	-1270387703112.833496
108	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, TOI%, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, ...)	-1271520580887.794434

	feature_idx	feature_names	avg_score
107	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, ...)	-1279357274413.480713
106	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 112, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, Maj, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, ...)	-1275061722657.73584
105	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 49, 50, 51, 52, 56, 57, 58, 60, 61, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, iBLK.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, ...)	-1274183929543.157715

	feature_idx	feature_names	avg_score
104	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 47, 49, 50, 51, 52, 56, 57, 58, 60, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, iDS, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, ...)	-1249311717191.822998
103	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 39, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 57, 58, 60, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iFF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, ...)	-1256781280252.428467
102	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 57, 58, 60, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iBLK, iTKA.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, ...)	-1242037473257.541992

	feature_idx	feature_names	avg_score
101	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 38, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 57, 60, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, ...)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iCF.1, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iTKA.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, Pace, ...)	-1258133286482.305664
100	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 57, 60, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 127, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iTKA.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, GA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, Pace, GS/G)	-1249681437425.579102
99	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 57, 60, 62, 63, 64, 68, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iTKA.1, BLK%, iFOW, iFOL, %FOT, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, Pace, GS/G)	-1242482385270.748047

	feature_idx	feature_names	avg_score
98	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 57, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 88, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Dflct, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, Pace, GS/G)	-1236732466111.773438
97	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 25, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 57, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, Pace, GS/G)	-1233166871990.906982
96	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 57, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, Pace, GS/G)	-1236022410395.260742

	feature_idx	feature_names	avg_score
95	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Pct%, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, Pace, GS/G)	-1232931752793.239746
94	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, OTOI, Grit, DAP, Pace, GS/G)	-1228711743175.205811
93	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 89, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Slap, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, Grit, DAP, Pace, GS/G)	-1223230565630.96582
92	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 141, 142, 143, 144, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, PEND, OPS, DPS, PS, Grit, DAP, Pace, GS/G)	-1219231389301.507324

	feature_idx	feature_names	avg_score
91	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 49, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, sDist.1, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, DAP, Pace, GS/G)	-1201039653965.979248
90	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 147, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, DAP, Pace, GS/G)	-1210043670906.282715
89	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Position, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1205288470320.556152
88	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 90, 93, 95, 98, 101, 104, 105, 106, 108, 110, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, NPD, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1211324135517.053711

	feature_idx	feature_names	avg_score
87	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 116, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, CF, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1198399485104.199951
86	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, PSA, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1206868779542.214844
85	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Hand, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1200364091154.67627
84	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, PSG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1189659337571.827393

	feature_idx	feature_names	avg_score
83	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, ENG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1192090159379.638184
82	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 43, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, ixG, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1191741645778.401367
81	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(Born, City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1189837980888.067383
80	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 148, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, Pace, GS/G)	-1184181950642.874512

	feature_idx	feature_names	avg_score
79	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 143, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, DPS, PS, Grit, GS/G)	-1173177892746.588135
78	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 70, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, dzFOL, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1168727855649.470703
77	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 72, 74, 75, 76, 77, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, nzFOL, ozFOL, FOW.Up, FOL.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1168174354689.474854
76	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 72, 74, 75, 77, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, nzFOL, ozFOL, FOW.Up, FOW.Down, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1159939287411.338623

	feature_idx	feature_names	avg_score
75	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 63, 64, 69, 72, 74, 75, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOW, iFOL, dzFOW, nzFOL, ozFOL, FOW.Up, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1164228940330.188965
74	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 15, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 69, 72, 74, 75, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, dzFOW, nzFOL, ozFOL, FOW.Up, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1158181277563.830566
73	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 69, 72, 74, 75, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 125, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, dzFOW, nzFOL, ozFOL, FOW.Up, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, SCA, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1153710820584.92041
72	(1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 69, 72, 74, 75, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, DftRd, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, dzFOW, nzFOL, ozFOL, FOW.Up, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1150021581421.572998
71	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 69, 72, 74, 75, 78, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, dzFOW, nzFOL, ozFOL, FOW.Up, FOL.Down, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1147123725485.763672

	feature_idx	feature_names	avg_score
70	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 69, 72, 74, 75, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, dzFOW, nzFOL, ozFOL, FOW.Up, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1145802495531.268555
69	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 37, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 69, 72, 75, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iCF, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, dzFOW, nzFOL, FOW.Up, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1142529549221.194824
68	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 69, 72, 75, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146, 150)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, dzFOW, nzFOL, FOW.Up, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit, GS/G)	-1139525606642.160156
67	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 69, 72, 75, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, dzFOW, nzFOL, FOW.Up, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit)	-1136849824216.029053
66	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 72, 75, 79, 80, 82, 83, 90, 93, 95, 98, 101, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, nzFOL, FOW.Up, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Snap, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit)	-1134264114709.875488

	feature_idx	feature_names	avg_score
65	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 72, 75, 79, 80, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, nzFOL, FOW.Up, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit)	-1131594635499.246582
64	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 80, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, FOL.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit)	-1129511411063.106201
63	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 29, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SH%, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit)	-1128111135710.978516
62	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 28, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, IPP%, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit)	-1126502223901.580078
61	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 144, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, PS, Grit)	-1124705704692.720947

	feature_idx	feature_names	avg_score
60	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 32, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, F/60, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, Grit)	-1123409360387.042969
59	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 111, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, Min, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, Grit)	-1122224520551.668457
58	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 118, 119, 120, 128, 129, 130, 131, 132, 133, 137, 138, 139, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, FF, FA, SF, RBF, RBA, RSF, RSA, DSF, DSA, HA, GVA, TKA, PENT, OPS, Grit)	-1121125966322.265625
57	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 105, 106, 108, 118, 119, 120, 128, 129, 130, 131, 133, 137, 138, 139, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenT, iPenD, iPEND, FF, FA, SF, RBF, RBA, RSF, RSA, DSA, HA, GVA, TKA, PENT, OPS, Grit)	-1120099939111.164551
56	(1, 2, 3, 4, 5, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 106, 108, 118, 119, 120, 128, 129, 130, 131, 133, 137, 138, 139, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Ht, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenD, iPEND, FF, FA, SF, RBF, RBA, RSF, RSA, DSA, HA, GVA, TKA, PENT, OPS, Grit)	-1114508388405.968262
55	(1, 2, 3, 4, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 106, 108, 118, 119, 120, 128, 129, 130, 131, 133, 137, 138, 139, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenD, iPEND, FF, FA, SF, RBF, RBA, RSF, RSA, DSA, HA, GVA, TKA, PENT, OPS, Grit)	-1113632529013.978027

	feature_idx	feature_names	avg_score
54	(1, 2, 3, 4, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 106, 108, 118, 119, 120, 129, 130, 131, 133, 137, 138, 139, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenD, iPEND, FF, FA, SF, RBA, RSF, RSA, DSA, HA, GVA, TKA, PENT, OPS, Grit)	-1107151833031.628906
53	(1, 2, 3, 4, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 106, 108, 118, 119, 120, 129, 131, 133, 137, 138, 139, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenD, iPEND, FF, FA, SF, RBA, RSA, DSA, HA, GVA, TKA, PENT, OPS, Grit)	-1096721682851.801025
52	(1, 2, 3, 4, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 104, 106, 108, 118, 119, 120, 129, 131, 133, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, S.Wrst, iPenD, iPEND, FF, FA, SF, RBA, RSA, DSA, HA, GVA, PENT, OPS, Grit)	-1093027734796.984375
51	(1, 2, 3, 4, 6, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 106, 108, 118, 119, 120, 129, 131, 133, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, Wt, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, iPenD, iPEND, FF, FA, SF, RBA, RSA, DSA, HA, GVA, PENT, OPS, Grit)	-1100562693912.278809
50	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 52, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 106, 108, 118, 119, 120, 129, 131, 133, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iHF.1, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, iPenD, iPEND, FF, FA, SF, RBA, RSA, DSA, HA, GVA, PENT, OPS, Grit)	-1098681793758.986084
49	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 19, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 106, 108, 118, 119, 120, 129, 131, 133, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, +/-, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, iPenD, iPEND, FF, FA, SF, RBA, RSA, DSA, HA, GVA, PENT, OPS, Grit)	-1097717429009.489868
48	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 106, 108, 118, 119, 120, 129, 131, 133, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, iPenD, iPEND, FF, FA, SF, RBA, RSA, DSA, HA, GVA, PENT, OPS, Grit)	-1097484130671.989624

	feature_idx	feature_names	avg_score
47	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 106, 108, 118, 119, 120, 129, 133, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, iPenD, iPEND, FF, FA, SF, RBA, DSA, HA, GVA, PENT, OPS, Grit)	-1097484130671.98999
46	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 106, 108, 118, 119, 120, 133, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, iPenD, iPEND, FF, FA, SF, DSA, HA, GVA, PENT, OPS, Grit)	-1089795797200.049072
45	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 106, 108, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, iPenD, iPEND, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1087357899730.032837
44	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 108, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, iPEND, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1087444478391.837158
43	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 51, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iHF, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1086318873180.546875
42	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 33, 36, 41, 42, 44, 45, 50, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, A/60, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1086457605926.253296
41	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 36, 41, 42, 44, 45, 50, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSF.2, iSCF, iRB, Pass, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1086596703947.589478
40	(1, 2, 3, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, Cntry, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1086859734911.962891

	feature_idx	feature_names	avg_score
39	(1, 2, 4, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, Nat, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1087719297359.922852
38	(1, 2, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 56, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iGVA, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1085994038016.416382
37	(1, 2, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 118, 119, 120, 137, 138, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, FF, FA, SF, HA, GVA, PENT, OPS, Grit)	-1085772314822.223389
36	(1, 2, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 64, 75, 79, 82, 83, 90, 93, 95, 98, 118, 119, 120, 137, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, G.Wrst, Post, S.Bkhd, FF, FA, SF, HA, PENT, OPS, Grit)	-1084276640594.962524
35	(1, 2, 7, 9, 12, 13, 16, 17, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 64, 75, 79, 82, 83, 90, 95, 98, 118, 119, 120, 137, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, A2, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, Post, S.Bkhd, FF, FA, SF, HA, PENT, OPS, Grit)	-1084979105353.438843
34	(1, 2, 7, 9, 12, 13, 16, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 64, 75, 79, 82, 83, 90, 95, 98, 118, 119, 120, 137, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, G.Snap, Post, S.Bkhd, FF, FA, SF, HA, PENT, OPS, Grit)	-1085486054065.777954
33	(1, 2, 7, 9, 12, 13, 16, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 64, 75, 79, 82, 83, 95, 98, 118, 119, 120, 137, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, Post, S.Bkhd, FF, FA, SF, HA, PENT, OPS, Grit)	-1085609416971.328369
32	(1, 2, 7, 9, 12, 13, 16, 20, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 64, 75, 79, 82, 83, 95, 98, 119, 120, 137, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, E+/-, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, Post, S.Bkhd, FA, SF, HA, PENT, OPS, Grit)	-1086632398434.630249
31	(1, 2, 7, 9, 12, 13, 16, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 64, 75, 79, 82, 83, 95, 98, 119, 120, 137, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, iFOL, FOW.Up, FOW.Close, 1G, GWG, Post, S.Bkhd, FA, SF, HA, PENT, OPS, Grit)	-1087822114752.250977
30	(1, 2, 7, 9, 12, 13, 16, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 75, 79, 82, 83, 95, 98, 119, 120, 137, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, FOW.Up, FOW.Close, 1G, GWG, Post, S.Bkhd, FA, SF, HA, PENT, OPS, Grit)	-1089383062841.330811

	feature_idx	feature_names	avg_score
29	(1, 2, 7, 9, 12, 13, 16, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 75, 79, 82, 83, 95, 98, 119, 120, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, FOW.Up, FOW.Close, 1G, GWG, Post, S.Bkhd, FA, SF, PENT, OPS, Grit)	-1090559905132.88147
28	(1, 2, 7, 9, 12, 13, 16, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 75, 79, 82, 83, 95, 98, 119, 140, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, FOW.Up, FOW.Close, 1G, GWG, Post, S.Bkhd, FA, PENT, OPS, Grit)	-1090851822548.717529
27	(1, 2, 7, 9, 12, 13, 16, 24, 26, 30, 31, 36, 41, 44, 45, 50, 60, 62, 75, 79, 82, 83, 95, 98, 119, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, TOIX, TOI/GP.1, SV%, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, FOW.Up, FOW.Close, 1G, GWG, Post, S.Bkhd, FA, OPS, Grit)	-1091751207867.980835
26	(1, 2, 7, 9, 12, 13, 16, 24, 26, 31, 36, 41, 44, 45, 50, 60, 62, 75, 79, 82, 83, 95, 98, 119, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, TOIX, TOI/GP.1, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, FOW.Up, FOW.Close, 1G, GWG, Post, S.Bkhd, FA, OPS, Grit)	-1093389616577.717041
25	(1, 2, 7, 9, 12, 13, 16, 24, 26, 31, 36, 41, 44, 45, 50, 60, 62, 75, 79, 83, 95, 98, 119, 142, 146)	(City, Pr/St, DftYr, Ovrl, Team, GP, A1, TOIX, TOI/GP.1, PDO, Diff/60, iSF.1, iSCF, iRB, Pass, iTKA.1, BLK%, FOW.Up, FOW.Close, GWG, Post, S.Bkhd, FA, OPS, Grit)	-1094957082864.455688

In [98]:
```
# Lets access the indices of the best features directly via the k_feature_idx_ attr
sfs2.k_feature_names_, sfs2.k_feature_idx_
```

Out[98]:
```
(('City',
  'Pr/St',
  'DftYr',
  'Ovrl',
  'Team',
  'GP',
  'A1',
  'TOIX',
  'TOI/GP.1',
  'PDO',
  'Diff/60',
  'iSF.1',
  'iSCF',
  'iRB',
  'Pass',
  'iTKA.1',
  'BLK%',
  'FOW.Up',
  'FOW.Close',
  'GWG',
  'Post',
  'S.Bkhd',
  'FA',
  'OPS',
  'Grit'),
 (1,
  2,
  7,
  9,
  12,
  13,
  16,
  24,
  26,
  31,
  36,
  41,
  44,
  45,
  50,
  60,
  62,
  75,
  79,
  83,
  95,
  98,
  119,
  142,
  146))
```

In [99]:
```
selected_feat_b= X.columns[list(sfs2.k_feature_idx_)]

print(selected_feat_b)
print("----------------------------------------------------")
print("Total Number of Selected Features: {}".format(len(selected_feat_b)))
```

```
Index(['City', 'Pr/St', 'DftYr', 'Ovrl', 'Team', 'GP', 'A1', 'TOIX', 'TOI/GP.1',
'PDO', 'Diff/60', 'iSF.1', 'iSCF', 'iRB', 'Pass', 'iTKA.1', 'BLK%', 'FOW.Up', 'FO
W.Close', 'GWG', 'Post', 'S.Bkhd', 'FA', 'OPS', 'Grit'], dtype='object')
----------------------------------------------------
Total Number of Selected Features: 25
```

Checking for best features out of all these above methods of feature selection

For Forward Selection

In [101…
```
from sklearn.pipeline import Pipeline

from sklearn.linear_model import LinearRegression
from sklearn. linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
from sklearn.preprocessing import StandardScaler
```

In [102…
```
X_train, X_test, y_train, y_test = train_test_split(data_final[selected_feat], data
                                                    test_size = 0.2,random_state=20
```

In [103…
```
lr=LinearRegression()
lr.fit(X_train,y_train)
# y_pred_linear=lr.predict(X_test)
```

Out[103]:
```
LinearRegression()
```

In [104…
```
# Evaluate the linear regression model using the 'r2_score', 'mean_squared_error' &
y_pred=lr.predict(X_train)
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
r_square=r2_score(y_train,y_pred)
print("R2=",r2_score(y_train,y_pred))
adjusted_r_square=1-(1-r_square)*(len(y_train)-1)/(len(y_train)-X_train.shape[1]-1
print("Adjusted_R2=",adjusted_r_square)
print("MSE=",mean_squared_error(y_train,y_pred))
print("MAE=",mean_absolute_error(y_train,y_pred))
```

```
R2= 0.7614642716587647
Adjusted_R2= 0.7484721295922487
MSE= 908410513668.7196
MAE= 719760.8512436545
```

For Backward Selection`

In [105…
```
X_train, X_test, y_train, y_test = train_test_split(data_final[selected_feat_b], da
                                                    test_size = 0.2,random_state=20
```

In [106…
```
lr=LinearRegression()
lr.fit(X_train,y_train)
# y_pred_linear=lr.predict(X_test)
```

Out[106]:
```
LinearRegression()
```

In [107…
```
# Evaluate the linear regression model using the 'r2_score', 'mean_squared_error' &
y_pred=lr.predict(X_train)
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
r_square=r2_score(y_train,y_pred)
print("R2=",r2_score(y_train,y_pred))
adjusted_r_square=1-(1-r_square)*(len(y_train)-1)/(len(y_train)-X_train.shape[1]-1
```

```
print("Adjusted_R2=",adjusted_r_square)
print("MSE=",mean_squared_error(y_train,y_pred))
print("MAE=",mean_absolute_error(y_train,y_pred))
```

```
R2= 0.7515470370806437
Adjusted_R2= 0.7380147406253411
MSE= 946178106053.8586
MAE= 731823.1307355042
```

Final Model Input Data using selected features

In [108…
```
data_final[selected_feat]
```

Out[108]:

	City	Pr/St	Wt	DftYr	Ovrl	Team	GP	Shifts	
0	-0.144318	0.238537	-0.721232	1.518691	-0.800156	0.073202	-1.753129	-1.534885	-1.467(
1	-0.986221	-0.047152	0.429238	0.755047	-0.854330	0.073202	0.938735	1.764529	1.686
2	2.090207	1.846980	1.173660	-0.772241	-0.998792	0.571275	0.455580	0.426929	0.649
3	-0.986221	-0.047152	1.309009	0.245951	-1.071023	-1.841273	-0.752307	-0.503217	-0.425
4	-0.527717	-0.047152	1.105985	0.755047	-0.836272	1.440450	1.042269	0.440648	0.350
…	…	…	…	…	…	…	…	…	
607	-1.435809	-2.811032	0.023190	0.500499	0.355540	0.648544	0.628136	0.173128	0.066
608	2.084231	1.869348	0.361563	-1.535884	-0.998792	0.764257	1.042269	1.628711	2.333
609	1.765921	0.238537	0.293889	-1.026788	-0.493174	2.047646	0.800691	1.166382	1.270
610	3.912785	-0.047152	-0.315183	-2.044980	0.156905	0.070262	0.904224	1.115622	0.786
611	-1.206781	-0.541160	1.715057	0.755047	0.427771	2.110751	1.007758	1.691819	1.485

607 rows × 25 columns

In [109…
```
model_input = pd.concat([data_final['Salary'],data_final[selected_feat]],axis=1)
```

In [110…
```
model_input
```

Out[110]:

	Salary	City	Pr/St	Wt	DftYr	Ovrl	Team	GP	Sh
0	925000.0	-0.144318	0.238537	-0.721232	1.518691	-0.800156	0.073202	-1.753129	-1.534
1	2250000.0	-0.986221	-0.047152	0.429238	0.755047	-0.854330	0.073202	0.938735	1.764
2	7636250.0	2.090207	1.846980	1.173660	-0.772241	-0.998792	0.571275	0.455580	0.426
3	3500000.0	-0.986221	-0.047152	1.309009	0.245951	-1.071023	-1.841273	-0.752307	-0.503
4	1750000.0	-0.527717	-0.047152	1.105985	0.755047	-0.836272	1.440450	1.042269	0.440
...	...	...	...	...	...	...	...	...	
607	600000.0	-1.435809	-2.811032	0.023190	0.500499	0.355540	0.648544	0.628136	0.173
608	7636250.0	2.084231	1.869348	0.361563	-1.535884	-0.998792	0.764257	1.042269	1.628
609	4250000.0	1.765921	0.238537	0.293889	-1.026788	-0.493174	2.047646	0.800691	1.166
610	7000000.0	3.912785	-0.047152	-0.315183	-2.044980	0.156905	0.070262	0.904224	1.115
611	925000.0	-1.206781	-0.541160	1.715057	0.755047	0.427771	2.110751	1.007758	1.691

607 rows × 26 columns

In [111…
```
model_input.shape
```

Out[111]:
```
(607, 26)
```

Dealing with Correlation

In [112…
```
correlation =data_final[selected_feat].corr()
correlation
```

Out[112]:

	City	Pr/St	Wt	DftYr	Ovrl	Team	GP	Shifts	
City	1.000000	0.260748	0.156800	-0.294362	-0.079070	0.100768	0.245281	0.278962	0.
Pr/St	0.260748	1.000000	0.084221	-0.169909	-0.035093	0.057481	0.107173	0.113806	0.
Wt	0.156800	0.084221	1.000000	-0.167709	-0.150561	-0.009768	0.140479	0.153386	0.
DftYr	-0.294362	-0.169909	-0.167709	1.000000	-0.162578	-0.167037	-0.301436	-0.292152	-0.
Ovrl	-0.079070	-0.035093	-0.150561	-0.162578	1.000000	-0.004676	-0.090853	-0.130347	-0.
Team	0.100768	0.057481	-0.009768	-0.167037	-0.004676	1.000000	0.111243	0.109634	0.
GP	0.245281	0.107173	0.140479	-0.301436	-0.090853	0.111243	1.000000	0.949884	0.
Shifts	0.278962	0.113806	0.153386	-0.292152	-0.130347	0.109634	0.949884	1.000000	0.
TOI	0.292591	0.114895	0.147265	-0.280845	-0.148188	0.109122	0.923837	0.990441	1.
TOI/GP	0.279655	0.097775	0.119815	-0.179345	-0.169686	0.062444	0.594453	0.776282	0.
iSCF	0.235055	0.084324	-0.024353	-0.157652	-0.134758	0.096258	0.610538	0.536156	0.
iHDf	-0.100174	0.039619	0.302122	-0.073412	0.082988	-0.107198	-0.078874	-0.155288	-0.
iTKA.1	0.227812	0.097741	-0.022190	-0.149415	-0.156243	0.098018	0.749668	0.750865	0.
BLK%	0.005041	-0.025895	0.165167	-0.080426	0.037138	0.030794	0.021231	0.130416	0.
1G	0.247239	0.084419	0.015425	-0.123939	-0.153369	0.120520	0.562797	0.532255	0.
GWG	0.213477	0.093245	-0.022766	-0.171989	-0.150696	0.148842	0.569655	0.574073	0.
G.Snap	0.213077	0.090842	0.003969	-0.087584	-0.152491	0.014023	0.491180	0.467292	0.
S.Bkhd	0.193900	0.078535	-0.009806	-0.148735	-0.134094	0.092724	0.623019	0.536340	0.
iPenD	0.163872	0.066748	-0.002449	-0.123525	-0.106161	0.086216	0.703575	0.660120	0.
FA	0.274829	0.110528	0.164873	-0.286799	-0.140855	0.081684	0.908176	0.978049	0.
xGF	0.324206	0.129736	0.103096	-0.245386	-0.174779	0.118002	0.852003	0.927403	0.
FOW	0.311680	0.133663	0.139190	-0.292770	-0.150986	0.093441	0.884896	0.970969	0.
FOL	0.310379	0.130786	0.147236	-0.284423	-0.151551	0.107583	0.898625	0.978858	0.
PEND	0.281529	0.117116	0.114528	-0.254254	-0.127419	0.086241	0.896280	0.956321	0.
OPS	0.280680	0.103340	-0.023629	-0.111578	-0.206213	0.148954	0.549694	0.604589	0.

In [113…

```
numerics = ['int64', 'float64']
num_columns =data_final[selected_feat].select_dtypes(include=numerics).columns
```

In [114…

```
for column1 in data_final[num_columns]:
     for column2 in data_final[num_columns]:
        if column1!=column2 and data_final[column1].corr(data_final[column2])>=0.9
             print(f"{column1} and {column2} are highly correlated")
```

```
GP and Shifts are highly correlated
GP and TOI are highly correlated
GP and FA are highly correlated
Shifts and GP are highly correlated
Shifts and TOI are highly correlated
Shifts and FA are highly correlated
Shifts and xGF are highly correlated
Shifts and FOW are highly correlated
Shifts and FOL are highly correlated
Shifts and PEND are highly correlated
TOI and GP are highly correlated
TOI and Shifts are highly correlated
TOI and FA are highly correlated
TOI and xGF are highly correlated
TOI and FOW are highly correlated
TOI and FOL are highly correlated
TOI and PEND are highly correlated
FA and GP are highly correlated
FA and Shifts are highly correlated
FA and TOI are highly correlated
FA and xGF are highly correlated
FA and FOW are highly correlated
FA and FOL are highly correlated
FA and PEND are highly correlated
xGF and Shifts are highly correlated
xGF and TOI are highly correlated
xGF and FA are highly correlated
xGF and FOW are highly correlated
xGF and FOL are highly correlated
xGF and PEND are highly correlated
FOW and Shifts are highly correlated
FOW and TOI are highly correlated
FOW and FA are highly correlated
FOW and xGF are highly correlated
FOW and FOL are highly correlated
FOW and PEND are highly correlated
FOL and Shifts are highly correlated
FOL and TOI are highly correlated
FOL and FA are highly correlated
FOL and xGF are highly correlated
FOL and FOW are highly correlated
FOL and PEND are highly correlated
PEND and Shifts are highly correlated
PEND and TOI are highly correlated
PEND and FA are highly correlated
PEND and xGF are highly correlated
PEND and FOW are highly correlated
PEND and FOL are highly correlated
```

In [115…]
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [116…]
```
def vif(Z3):
    vif_data = pd.DataFrame()
    vif_data["feature"] = Z3.columns
    vif_data["VIF"] = [variance_inflation_factor(np.array(Z3), i)
    for i in range(len(Z3.columns))]
    return vif_data
```

In [117…]
```
vif_df=vif(data_final[num_columns])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[117]:

	feature	VIF
8	TOI	275.997627
7	Shifts	120.556201
22	FOL	67.006554
19	FA	53.410172
21	FOW	41.798281
20	xGF	39.573769
6	GP	36.421730
23	PEND	21.915678
10	iSCF	10.001090
24	OPS	8.183473
9	TOI/GP	7.699131
17	S.Bkhd	4.836816
12	iTKA.1	4.076159
15	GWG	3.872114
18	iPenD	3.168959
14	1G	2.937816
16	G.Snap	2.549344
13	BLK%	1.644137
11	iHDf	1.420481
3	DftYr	1.362854
0	City	1.328285
2	Wt	1.300892
4	Ovrl	1.169545
5	Team	1.135952
1	Pr/St	1.115348

In [118…
```
len(num_columns)
```

Out[118]:
```
25
```

In [119…
```
vif_list=['City', 'Pr/St', 'Wt', 'DftYr', 'Ovrl', 'Team', 'GP', 'Shifts', 'TOI', '
```

In [120…
```
vif_list=[i for i in vif_list if i not in ['TOI']]
vif_list
```

Out[120]:
```
['City',
 'Pr/St',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'Shifts',
 'TOI/GP',
 'iSCF',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD',
 'FA',
 'xGF',
 'FOW',
 'FOL',
 'PEND',
 'OPS']
```

In [121…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[121]:

	feature	VIF
7	Shifts	104.846593
21	FOL	62.840144
18	FA	43.688123
20	FOW	39.665056
6	GP	30.928198
19	xGF	30.224547
22	PEND	21.525675
9	iSCF	8.866232
23	OPS	8.181380
8	TOI/GP	6.942099
16	S.Bkhd	4.834312
11	iTKA.1	4.005743
14	GWG	3.863850
17	iPenD	3.166520
13	1G	2.933595
15	G.Snap	2.549159
12	BLK%	1.638493
10	iHDf	1.401209
3	DftYr	1.362115
0	City	1.328090
2	Wt	1.299052
4	Ovrl	1.165493
5	Team	1.127007
1	Pr/St	1.105148

In [122…

```
vif_list=[i for i in vif_list if i not in ['Shifts']]
vif_list
```

Out[122]:
```
['City',
 'Pr/St',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'TOI/GP',
 'iSCF',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD',
 'FA',
 'xGF',
 'FOW',
 'FOL',
 'PEND',
 'OPS']
```

In [123…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[123]:

	feature	VIF
20	FOL	58.111059
17	FA	40.718869
19	FOW	34.015201
18	xGF	30.189943
21	PEND	21.524265
6	GP	15.938703
8	iSCF	8.818242
22	OPS	8.105825
7	TOI/GP	6.757741
15	S.Bkhd	4.738968
10	iTKA.1	4.005027
13	GWG	3.855463
16	iPenD	3.165897
12	1G	2.932288
14	G.Snap	2.549101
11	BLK%	1.624158
9	iHDf	1.366050
3	DftYr	1.357532
0	City	1.318309
2	Wt	1.289782
4	Ovrl	1.164911
5	Team	1.119783
1	Pr/St	1.103166

In [124…

```
vif_list=[i for i in vif_list if i not in ['FOL']]
vif_list
```

Out[124]:
```
['City',
 'Pr/St',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'TOI/GP',
 'iSCF',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD',
 'FA',
 'xGF',
 'FOW',
 'PEND',
 'OPS']
```

In [125…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[125]:

	feature	VIF
19	FOW	32.202564
18	xGF	28.920616
17	FA	27.803064
20	PEND	20.141996
6	GP	15.935263
8	iSCF	8.803315
21	OPS	8.096398
7	TOI/GP	6.690471
15	S.Bkhd	4.732480
10	iTKA.1	3.979429
13	GWG	3.804130
16	iPenD	3.157897
12	1G	2.931876
14	G.Snap	2.521729
11	BLK%	1.623262
9	iHDf	1.365909
3	DftYr	1.357485
0	City	1.307583
2	Wt	1.287928
4	Ovrl	1.164908
5	Team	1.112563
1	Pr/St	1.101019

In [126…

```
vif_list=[i for i in vif_list if i not in ['FOW']]
vif_list
```

Out[126]:
```
['City',
 'Pr/St',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'TOI/GP',
 'iSCF',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD',
 'FA',
 'xGF',
 'PEND',
 'OPS']
```

In [127…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[127]:

	feature	VIF
18	xGF	24.872395
17	FA	21.512493
19	PEND	18.381305
6	GP	15.808140
8	iSCF	8.778745
20	OPS	8.049275
7	TOI/GP	6.630521
15	S.Bkhd	4.689761
10	iTKA.1	3.977234
13	GWG	3.791971
16	iPenD	3.141985
12	1G	2.931300
14	G.Snap	2.475734
11	BLK%	1.623230
9	iHDf	1.365904
3	DftYr	1.337728
0	City	1.305852
2	Wt	1.287910
4	Ovrl	1.164739
5	Team	1.111045
1	Pr/St	1.098112

In [128…
```
vif_list=[i for i in vif_list if i not in ['xGF']]
vif_list
```

Out[128]:
```
['City',
 'Pr/St',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'TOI/GP',
 'iSCF',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD',
 'FA',
 'PEND',
 'OPS']
```

In [129…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[129]:

	feature	VIF
17	FA	20.537589
6	GP	15.574102
18	PEND	15.272579
8	iSCF	7.783062
19	OPS	6.007628
7	TOI/GP	5.969391
15	S.Bkhd	4.601800
10	iTKA.1	3.966797
13	GWG	3.680174
16	iPenD	3.025703
12	1G	2.879215
14	G.Snap	2.465155
11	BLK%	1.622793
9	iHDf	1.365887
3	DftYr	1.337707
0	City	1.303996
2	Wt	1.287067
4	Ovrl	1.164441
5	Team	1.107586
1	Pr/St	1.097405

In [130…
```
vif_list=[i for i in vif_list if i not in ['FA']]
vif_list
```

Out[130]:
```
['City',
 'Pr/St',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'TOI/GP',
 'iSCF',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD',
 'PEND',
 'OPS']
```

In [131…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[131]:

	feature	VIF
17	PEND	13.748509
6	GP	9.159228
8	iSCF	7.556801
18	OPS	5.980288
15	S.Bkhd	4.587164
7	TOI/GP	4.104309
10	iTKA.1	3.957154
13	GWG	3.680103
16	iPenD	2.969065
12	1G	2.874940
14	G.Snap	2.464712
11	BLK%	1.622770
9	iHDf	1.365867
3	DftYr	1.331072
0	City	1.303978
2	Wt	1.285411
4	Ovrl	1.152072
5	Team	1.104578
1	Pr/St	1.097072

In [133…
```
vif_list=[i for i in vif_list if i not in ['PEND']]
vif_list
```

Out[133]:
```
['City',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'TOI/GP',
 'iSCF',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD',
 'OPS']
```

In [134…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[134]:

	feature	VIF
7	iSCF	7.373752
16	OPS	5.366951
14	S.Bkhd	4.575840
5	GP	3.974844
9	iTKA.1	3.947314
12	GWG	3.679559
6	TOI/GP	2.909129
11	1G	2.827979
15	iPenD	2.698012
13	G.Snap	2.463593
10	BLK%	1.619179
8	iHDf	1.359199
2	DftYr	1.320864
1	Wt	1.284763
0	City	1.246714
3	Ovrl	1.149701
4	Team	1.093893

In [135…
```
vif_list=[i for i in vif_list if i not in ['iSCF']]
vif_list
```

Out[135]:
```
['City',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'TOI/GP',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD',
 'OPS']
```

In [136…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[136]:

	feature	VIF
15	OPS	5.207965
5	GP	3.967193
8	iTKA.1	3.867485
11	GWG	3.540981
13	S.Bkhd	3.150363
10	1G	2.791939
6	TOI/GP	2.779160
14	iPenD	2.672351
12	G.Snap	2.364127
9	BLK%	1.577210
7	iHDf	1.355308
2	DftYr	1.317626
1	Wt	1.284683
0	City	1.239357
3	Ovrl	1.149441
4	Team	1.093253

In [137…
```
vif_list=[i for i in vif_list if i not in ['OPS']]
vif_list
```

Out[137]:
```
['City',
 'Wt',
 'DftYr',
 'Ovrl',
 'Team',
 'GP',
 'TOI/GP',
 'iHDf',
 'iTKA.1',
 'BLK%',
 '1G',
 'GWG',
 'G.Snap',
 'S.Bkhd',
 'iPenD']
```

In [138…
```
vif_df=vif(data_final[vif_list])
vif_df.sort_values(by=['VIF'],ascending=False)
```

Out[138]:

	feature	VIF
8	iTKA.1	3.766578
5	GP	3.720078
13	S.Bkhd	3.128672
11	GWG	2.898886
14	iPenD	2.659686
10	1G	2.559404
6	TOI/GP	2.341981
12	G.Snap	2.195999
9	BLK%	1.569712
7	iHDf	1.352085
2	DftYr	1.315083
1	Wt	1.284126
0	City	1.233141
3	Ovrl	1.147356
4	Team	1.080501

In [139…
```
len(vif_list)
```

Out[139]:
```
15
```

In [140…
```
model_input = pd.concat([data_final['Salary'],data_final[vif_list]],axis=1)
```

Train test split of data

In [141…
```
# Use numpy to convert to arrays
import numpy as np
# Labels are the values we want to predict
labels = np.array(model_input['Salary'])
# Remove the labels from the features
```

```
# axis 1 refers to the columns
features= model_input.drop('Salary', axis = 1)
# Saving feature names for later use
feature_list = list(features.columns)
# Convert to numpy array
features = np.array(features)
```

In [142…
```
# Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
train_features, test_features, train_labels, test_labels = train_test_split(feature
```

In [143…
```
print('Training Features Shape:', train_features.shape)
print('Training Labels Shape:', train_labels.shape)
print('Testing Features Shape:', test_features.shape)
print('Testing Labels Shape:', test_labels.shape)
```

```
Training Features Shape: (455, 15)
Training Labels Shape: (455,)
Testing Features Shape: (152, 15)
Testing Labels Shape: (152,)
```

In [144…
```
test_labels
```

Out[144]:
```
array([1025000.,  667500.,  667500., 3900000., 2500000.,  600000.,
        800000., 4000000.,  792500.,  925000.,  650000., 3250000.,
        925000.,  775000.,  600000.,  817500., 6000000.,  692500.,
       2500000., 5000000., 7636250.,  900000.,  600000., 5000000.,
        700000.,  800000., 4000000.,  840000.,  832500.,  575000.,
        667500., 5750000., 1000000.,  615000., 1700000.,  925000.,
       3700000.,  792500., 3100000.,  900000.,  650000.,  600000.,
        600000.,  850000.,  600000.,  717500., 6000000., 3400000.,
        575000., 4500000., 5000000., 2950000., 2725000.,  925000.,
       1000000.,  874125.,  575000., 5000000., 2900000., 7636250.,
       3250000.,  667500., 5000000.,  750000.,  700000., 1500000.,
       1000000., 6000000.,  875000.,  842500., 1300000., 1850000.,
        950000.,  575000.,  900000.,  832500.,  650000., 1000000.,
       3500000., 3275000.,  800000.,  925000.,  575000.,  635000.,
        575000.,  925000., 4500000., 2500000.,  900000.,  600000.,
       4250000., 4500000.,  600000.,  767500., 3500000.,  832500.,
        832500.,  600000.,  600000., 7636250., 5500000., 3250000.,
        950000., 4000000., 2600000.,  625000.,  650000., 5000000.,
       4000000., 4000000., 1300000., 3500000., 7636250.,  925000.,
        667500., 4875000.,  925000., 6000000.,  575000., 1050000.,
       2000000.,  900000.,  925000.,  832500.,  925000., 4250000.,
        575000., 2000000., 4500000., 4000000.,  625000.,  575000.,
       4250000., 4500000., 5750000.,  640000.,  832500., 7000000.,
        825000., 3550000., 6750000.,  680000.,  600000.,  800000.,
        715000., 4000000.,  750000., 6500000., 7500000.,  725000.,
        925000.,  575000.])
```

linear regression

In [153…
```
### Linear regression
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, mean_absolute_error
lr_model = linear_model.LinearRegression()
lr_model.fit(train_features, train_labels)
y_pred_lr = lr_model.predict(test_features)
mse_lr = mean_squared_error(test_labels, y_pred_lr)
mae_lr = mean_absolute_error(test_labels, y_pred_lr)
print('Mean squared error from linear regression: ', mse_lr)
```

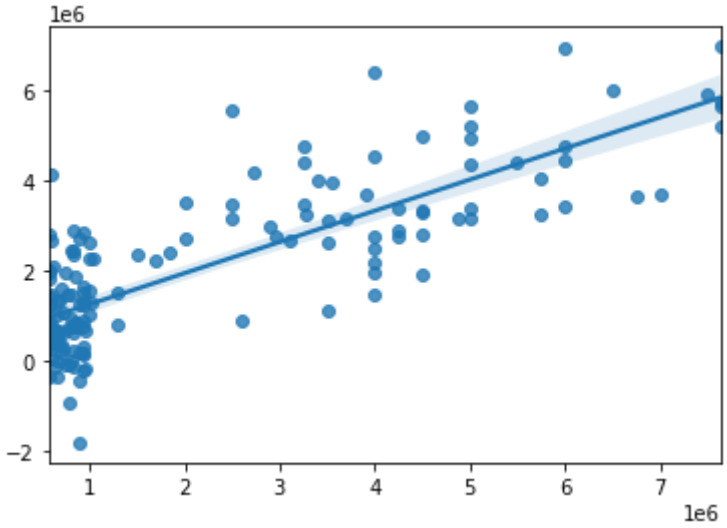
```
print('Mean absolute error from linear regression: ', mae_lr)
r_square=r2_score(test_labels,y_pred_lr)
print("R2=",r2_score(test_labels,y_pred_lr))
adjusted_r_square=1-(1-r_square)*(len(test_labels)-1)/(len(test_labels)-test_featu
print("Adjusted_R2=",adjusted_r_square)
y_pred_lr
```

```
Mean squared error from linear regression:  1527608871616.554
Mean absolute error from linear regression:  950393.348259453
R2= 0.6298781218534252
Adjusted_R2= 0.5890558558813765
```

Out[153]:
```
array([ 1.27105506e+06,  1.33873455e+06,  6.49907948e+05,  3.67366082e+06,
        3.15323724e+06,  3.41447352e+05,  9.38681904e+05,  6.40218450e+06,
       -9.27984779e+05,  2.97251199e+05, -2.15304635e+04,  4.73274447e+06,
        7.35420937e+05,  6.91940569e+05,  6.60689504e+05, -4.05343839e+03,
        4.74906074e+06,  1.64867366e+05,  3.45991310e+06,  4.94163767e+06,
        5.20459589e+06,  1.23950310e+06,  7.07900998e+05,  5.63129668e+06,
        3.74111990e+05,  2.44719490e+06,  1.93481905e+06,  8.52731671e+05,
        2.87917505e+06, -3.61063110e+05,  1.00238240e+06,  4.04428604e+06,
        1.54807246e+06,  2.08903603e+06,  2.20023069e+06,  2.85583973e+06,
        3.14403211e+06, -7.41719291e+04,  2.65169016e+06, -4.48984207e+05,
        6.52941762e+05,  2.77522857e+05,  1.47885801e+06,  1.87732573e+06,
        8.02147660e+05,  1.05123200e+06,  4.43052081e+06,  3.97027494e+06,
        1.17590429e+06,  3.30193752e+06,  3.14662373e+06,  2.75255874e+06,
        4.18439923e+06,  1.60906189e+05,  1.02181151e+06,  7.45694541e+05,
        1.93514907e+06,  3.36774903e+06,  2.95840610e+06,  5.65387384e+06,
        4.37144331e+06,  5.75024738e+05,  5.19760249e+06,  1.95461568e+06,
        1.58525168e+06,  2.33406442e+06,  2.26045373e+06,  3.40159306e+06,
        8.57878542e+05,  7.45392985e+05,  7.82715280e+05,  2.40434040e+06,
       -1.71494784e+05,  1.40203017e+06, -1.81666563e+06,  7.97440473e+05,
       -3.69346236e+05,  2.62567332e+06,  2.61488532e+06,  3.21932960e+06,
        1.45610640e+06,  1.32470447e+06,  1.21918608e+06,  6.35529703e+05,
        2.00251771e+06,  1.62919145e+06,  3.33701371e+06,  5.56166335e+06,
        1.56418831e+05,  2.65617982e+06,  3.36304916e+06,  4.97052829e+06,
        7.60381383e+05,  1.48241620e+06,  1.11438817e+06,  2.33194889e+06,
        6.04011375e+05,  4.13574553e+06,  5.22205078e+04,  6.95189185e+06,
        4.40672941e+06,  3.46958170e+06,  6.52415093e+05,  4.53875599e+06,
        9.00044900e+05,  5.03033658e+05,  6.07587622e+05,  4.32669894e+06,
        2.76137747e+06,  1.45330384e+06,  1.51266758e+06,  3.10049615e+06,
        5.69919381e+06,  1.23611647e+06,  4.33754743e+05,  3.16097948e+06,
        1.49360778e+06,  6.93296772e+06,  8.13424176e+05,  2.25928809e+06,
        2.70416373e+06,  2.68741533e+06,  1.33159984e+05,  2.23649799e+05,
        8.82916664e+05,  2.86953797e+06,  2.77269756e+06,  3.49368698e+06,
        1.91739579e+06,  2.47000889e+06,  4.10029745e+05,  2.00830717e+06,
        2.74308342e+06,  2.81413763e+06,  3.21702821e+06, -1.30561443e+05,
       -1.15503178e+05,  3.66341106e+06,  2.45128187e+06,  3.92652542e+06,
        3.63866350e+06,  1.31885929e+06,  5.60090611e+05,  7.73438195e+05,
        6.30701850e+05,  2.15481924e+06, -8.93745545e+04,  5.98190341e+06,
        5.90062229e+06,  2.72594162e+05, -2.11278835e+05,  1.83585393e+06])
```

In [146…
```
sns.regplot(test_labels, y_pred_lr)
```

Out[146]:
```
<AxesSubplot:>
```



ridge

In [196…

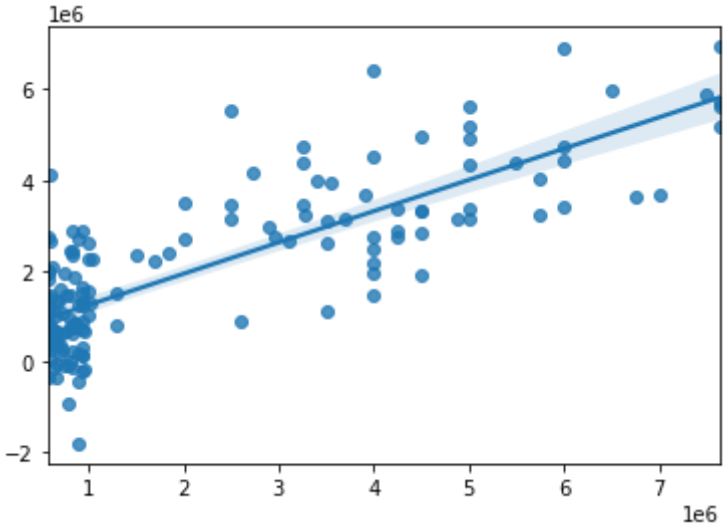
```
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score

pipeline = make_pipeline(StandardScaler(), Ridge(alpha=1.0))
pipeline.fit(train_features,train_labels )
#
# Calculate the predicted value for training and test dataset
#
y_train_pred = pipeline.predict(train_features)
y_test_pred_ridge = pipeline.predict(test_features)
#
# Mean Squared Error
#
print('MSE train: %.3f, test: %.3f' % (mean_squared_error(train_labels, y_train_pr
                mean_squared_error(test_labels, y_test_pred_ridge)))
#Mean Absolute error
print('MAE train: %.3f, test: %.3f' % (mean_absolute_error(train_labels, y_train_p
                mean_absolute_error(test_labels, y_test_pred_ridge)))
# R-Squared
r_square=r2_score(test_labels,y_test_pred_ridge)
print("R2=",r2_score(test_labels,y_test_pred_ridge))
#Adjusted r square
adjusted_r_square=1-(1-r_square)*(len(test_labels)-1)/(len(test_labels)-test_featu
print("Adjusted_R2=",adjusted_r_square)
y_test_pred_ridge
```

```
MSE train: 1153867092481.426, test: 1525057220169.325
MAE train: 837702.408, test: 949620.815
R2= 0.6304963573478448
Adjusted_R2= 0.5897422791141512
```

Out[196]:
```
array([ 1.26791043e+06,  1.33928593e+06,  6.47774221e+05,  3.67136377e+06,
        3.15868585e+06,  3.40275290e+05,  9.42718626e+05,  6.40053162e+06,
       -9.17415096e+05,  3.00413598e+05, -2.10734427e+04,  4.72755010e+06,
        7.31922081e+05,  6.88813918e+05,  6.61448670e+05, -6.47693638e+03,
        4.74034265e+06,  1.65550347e+05,  3.45759463e+06,  4.93724534e+06,
        5.20043055e+06,  1.23938589e+06,  7.14409819e+05,  5.63210597e+06,
        3.74092668e+05,  2.44572624e+06,  1.93599848e+06,  8.58324824e+05,
        2.86474471e+06, -3.57906279e+05,  1.00660986e+06,  4.03922828e+06,
        1.54972684e+06,  2.08030244e+06,  2.20474706e+06,  2.86019713e+06,
        3.14700757e+06, -6.26007952e+04,  2.65389591e+06, -4.52667839e+05,
        6.53977001e+05,  2.82068336e+05,  1.48628826e+06,  1.87658136e+06,
        8.05385672e+05,  1.05910392e+06,  4.42019456e+06,  3.96780149e+06,
        1.17059806e+06,  3.30306325e+06,  3.14598181e+06,  2.75249702e+06,
        4.16657819e+06,  1.57508404e+05,  1.02251882e+06,  7.48377405e+05,
        1.93077873e+06,  3.36015313e+06,  2.95919505e+06,  5.64495862e+06,
        4.37225045e+06,  5.68550671e+05,  5.19082543e+06,  1.95667965e+06,
        1.57869656e+06,  2.33698085e+06,  2.26233630e+06,  3.39764656e+06,
        8.61246146e+05,  7.45861134e+05,  7.84682384e+05,  2.40551514e+06,
       -1.70307721e+05,  1.40516690e+06, -1.80568855e+06,  8.10976005e+05,
       -3.67120869e+05,  2.61975699e+06,  2.61548745e+06,  3.22525489e+06,
        1.45643224e+06,  1.32811388e+06,  1.21739299e+06,  6.35329128e+05,
        1.99641666e+06,  1.63062804e+06,  3.32843888e+06,  5.55113935e+06,
        1.60367564e+05,  2.64900106e+06,  3.36326603e+06,  4.97086907e+06,
        7.63059562e+05,  1.47846966e+06,  1.11654372e+06,  2.32755653e+06,
        6.07013517e+05,  4.12650647e+06,  6.01376770e+04,  6.94004178e+06,
        4.40353830e+06,  3.46015545e+06,  6.52342955e+05,  4.53459596e+06,
        9.06329016e+05,  5.11643608e+05,  6.06725062e+05,  4.33022059e+06,
        2.75990051e+06,  1.45736286e+06,  1.51357254e+06,  3.10124125e+06,
        5.69830254e+06,  1.23871288e+06,  4.39537760e+05,  3.15811841e+06,
        1.49672836e+06,  6.92321000e+06,  8.22817483e+05,  2.26019877e+06,
        2.70415116e+06,  2.68962510e+06,  1.31497718e+05,  2.23614611e+05,
        8.87837537e+05,  2.87325241e+06,  2.76827222e+06,  3.49005024e+06,
        1.91773652e+06,  2.47354593e+06,  4.09083366e+05,  2.00177333e+06,
        2.74148451e+06,  2.81704338e+06,  3.21998324e+06, -1.27556655e+05,
       -1.07949249e+05,  3.66039304e+06,  2.45050895e+06,  3.92120467e+06,
        3.63663092e+06,  1.31662399e+06,  5.59797293e+05,  7.71605280e+05,
        6.33538133e+05,  2.15451220e+06, -8.22378658e+04,  5.97828585e+06,
        5.88955964e+06,  2.75996745e+05, -2.10070298e+05,  1.83236294e+06])
```

In [198…
```
sns.regplot(test_labels, y_test_pred_ridge)
```

Out[198]:
```
<AxesSubplot:>
```



Ann

In [199…
```
from pandas import read_csv
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
```

In [200…
```
# define the model
#Experiment with deeper and wider networks
model_ann = Sequential()
model_ann.add(Dense(128, input_dim=15, activation='relu'))
model_ann.add(Dense(64, activation='relu'))
#Output layer
model_ann.add(Dense(1, activation='linear'))

model_ann.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae'])
model_ann.summary()

history = model_ann.fit(train_features, train_labels, validation_split=0.2, epochs
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 128)               2048

 dense_7 (Dense)             (None, 64)                8256

 dense_8 (Dense)             (None, 1)                 65

=================================================================
Total params: 10,369
Trainable params: 10,369
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
12/12 [==============================] - 1s 14ms/step - loss: 8429216202752.0000 -
mae: 2104153.7500 - val_loss: 11013271322624.0000 - val_mae: 2604166.0000
Epoch 2/100
12/12 [==============================] - 0s 4ms/step - loss: 8429208338432.0000 -
mae: 2104152.0000 - val_loss: 11013256642560.0000 - val_mae: 2604163.5000
Epoch 3/100
12/12 [==============================] - 0s 4ms/step - loss: 8429193134080.0000 -
mae: 2104149.5000 - val_loss: 11013233573888.0000 - val_mae: 2604159.7500
Epoch 4/100
12/12 [==============================] - 0s 4ms/step - loss: 8429172162560.0000 -
mae: 2104145.0000 - val_loss: 11013192679424.0000 - val_mae: 2604153.7500
Epoch 5/100
12/12 [==============================] - 0s 4ms/step - loss: 8429136510976.0000 -
mae: 2104139.0000 - val_loss: 11013129764864.0000 - val_mae: 2604145.0000
Epoch 6/100
12/12 [==============================] - 0s 4ms/step - loss: 8429081460736.0000 -
mae: 2104129.7500 - val_loss: 11013039587328.0000 - val_mae: 2604131.7500
Epoch 7/100
12/12 [==============================] - 0s 4ms/step - loss: 8429000720384.0000 -
mae: 2104117.0000 - val_loss: 11012909563904.0000 - val_mae: 2604113.5000
Epoch 8/100
12/12 [==============================] - 0s 4ms/step - loss: 8428891144192.0000 -
mae: 2104099.0000 - val_loss: 11012716625920.0000 - val_mae: 2604087.2500
Epoch 9/100
12/12 [==============================] - 0s 4ms/step - loss: 8428733857792.0000 -
mae: 2104075.2500 - val_loss: 11012458676224.0000 - val_mae: 2604052.2500
Epoch 10/100
12/12 [==============================] - 0s 4ms/step - loss: 8428519424000.0000 -
mae: 2104043.5000 - val_loss: 11012126277632.0000 - val_mae: 2604007.5000
Epoch 11/100
12/12 [==============================] - 0s 4ms/step - loss: 8428252561408.0000 -
mae: 2104002.7500 - val_loss: 11011698458624.0000 - val_mae: 2603950.2500
Epoch 12/100
12/12 [==============================] - 0s 4ms/step - loss: 8427898142720.0000 -
mae: 2103952.2500 - val_loss: 11011178364928.0000 - val_mae: 2603881.0000
Epoch 13/100
12/12 [==============================] - 0s 4ms/step - loss: 8427473469440.0000 -
mae: 2103889.5000 - val_loss: 11010511470592.0000 - val_mae: 2603793.0000
Epoch 14/100
12/12 [==============================] - 0s 4ms/step - loss: 8426960191488.0000 -
mae: 2103814.7500 - val_loss: 11009696727040.0000 - val_mae: 2603685.7500
Epoch 15/100
12/12 [==============================] - 0s 5ms/step - loss: 8426329997312.0000 -
mae: 2103724.7500 - val_loss: 11008746717184.0000 - val_mae: 2603560.5000
Epoch 16/100
12/12 [==============================] - 0s 5ms/step - loss: 8425589178368.0000 -
mae: 2103619.0000 - val_loss: 11007635226624.0000 - val_mae: 2603414.7500
Epoch 17/100
```

```
12/12 [==============================] - 0s 4ms/step - loss: 8424730394624.0000 -
mae: 2103496.0000 - val_loss: 11006325555200.0000 - val_mae: 2603243.5000
Epoch 18/100
12/12 [==============================] - 0s 4ms/step - loss: 8423726383104.0000 -
mae: 2103353.0000 - val_loss: 11004835528704.0000 - val_mae: 2603049.5000
Epoch 19/100
12/12 [==============================] - 0s 4ms/step - loss: 8422586580992.0000 -
mae: 2103194.0000 - val_loss: 11003121106944.0000 - val_mae: 2602826.5000
Epoch 20/100
12/12 [==============================] - 0s 4ms/step - loss: 8421279006720.0000 -
mae: 2103008.5000 - val_loss: 11001238913024.0000 - val_mae: 2602581.7500
Epoch 21/100
12/12 [==============================] - 0s 4ms/step - loss: 8419803136000.0000 -
mae: 2102806.2500 - val_loss: 10999087235072.0000 - val_mae: 2602303.2500
Epoch 22/100
12/12 [==============================] - 0s 4ms/step - loss: 8418209824768.0000 -
mae: 2102580.0000 - val_loss: 10996593721344.0000 - val_mae: 2601981.2500
Epoch 23/100
12/12 [==============================] - 0s 5ms/step - loss: 8416371671040.0000 -
mae: 2102320.2500 - val_loss: 10993828626432.0000 - val_mae: 2601624.0000
Epoch 24/100
12/12 [==============================] - 0s 4ms/step - loss: 8414314364928.0000 -
mae: 2102037.7500 - val_loss: 10990929313792.0000 - val_mae: 2601249.5000
Epoch 25/100
12/12 [==============================] - 0s 4ms/step - loss: 8412145385472.0000 -
mae: 2101735.5000 - val_loss: 10987812945920.0000 - val_mae: 2600845.7500
Epoch 26/100
12/12 [==============================] - 0s 4ms/step - loss: 8409771409408.0000 -
mae: 2101411.7500 - val_loss: 10984399831040.0000 - val_mae: 2600406.0000
Epoch 27/100
12/12 [==============================] - 0s 4ms/step - loss: 8407194009600.0000 -
mae: 2101051.5000 - val_loss: 10980537925632.0000 - val_mae: 2599908.0000
Epoch 28/100
12/12 [==============================] - 0s 4ms/step - loss: 8404320911360.0000 -
mae: 2100656.0000 - val_loss: 10976441139200.0000 - val_mae: 2599379.5000
Epoch 29/100
12/12 [==============================] - 0s 4ms/step - loss: 8401181474816.0000 -
mae: 2100225.0000 - val_loss: 10972047605760.0000 - val_mae: 2598814.0000
Epoch 30/100
12/12 [==============================] - 0s 4ms/step - loss: 8398003765248.0000 -
mae: 2099777.0000 - val_loss: 10967082598400.0000 - val_mae: 2598176.2500
Epoch 31/100
12/12 [==============================] - 0s 4ms/step - loss: 8394486841344.0000 -
mae: 2099285.7500 - val_loss: 10961758978048.0000 - val_mae: 2597493.5000
Epoch 32/100
12/12 [==============================] - 0s 4ms/step - loss: 8390439862272.0000 -
mae: 2098738.7500 - val_loss: 10956348325888.0000 - val_mae: 2596797.0000
Epoch 33/100
12/12 [==============================] - 0s 4ms/step - loss: 8386440069120.0000 -
mae: 2098189.2500 - val_loss: 10950305382400.0000 - val_mae: 2596022.5000
Epoch 34/100
12/12 [==============================] - 0s 4ms/step - loss: 8381994631168.0000 -
mae: 2097575.7500 - val_loss: 10944038043648.0000 - val_mae: 2595219.0000
Epoch 35/100
12/12 [==============================] - 0s 4ms/step - loss: 8377317982208.0000 -
mae: 2096949.7500 - val_loss: 10937498075136.0000 - val_mae: 2594378.2500
Epoch 36/100
12/12 [==============================] - 0s 4ms/step - loss: 8372549582848.0000 -
mae: 2096286.7500 - val_loss: 10930536579072.0000 - val_mae: 2593485.2500
Epoch 37/100
12/12 [==============================] - 0s 4ms/step - loss: 8367256895488.0000 -
mae: 2095581.8750 - val_loss: 10923284627456.0000 - val_mae: 2592555.2500
Epoch 38/100
12/12 [==============================] - 0s 4ms/step - loss: 8361692102656.0000 -
```

```
mae: 2094819.8750 - val_loss: 10915143483392.0000 - val_mae: 2591512.5000
Epoch 39/100
12/12 [==============================] - 0s 4ms/step - loss: 8355888234496.0000 -
mae: 2094012.5000 - val_loss: 10906499022848.0000 - val_mae: 2590403.2500
Epoch 40/100
12/12 [==============================] - 0s 4ms/step - loss: 8349675421696.0000 -
mae: 2093179.3750 - val_loss: 10897958371328.0000 - val_mae: 2589308.5000
Epoch 41/100
12/12 [==============================] - 0s 4ms/step - loss: 8343434297344.0000 -
mae: 2092314.0000 - val_loss: 10888923840512.0000 - val_mae: 2588150.7500
Epoch 42/100
12/12 [==============================] - 0s 4ms/step - loss: 8336965107712.0000 -
mae: 2091400.1250 - val_loss: 10878999068672.0000 - val_mae: 2586879.0000
Epoch 43/100
12/12 [==============================] - 0s 7ms/step - loss: 8329609871360.0000 -
mae: 2090435.5000 - val_loss: 10869416132608.0000 - val_mae: 2585647.0000
Epoch 44/100
12/12 [==============================] - 0s 4ms/step - loss: 8322431320064.0000 -
mae: 2089459.0000 - val_loss: 10858891575296.0000 - val_mae: 2584296.5000
Epoch 45/100
12/12 [==============================] - 0s 4ms/step - loss: 8314770423808.0000 -
mae: 2088417.3750 - val_loss: 10848112214016.0000 - val_mae: 2582914.0000
Epoch 46/100
12/12 [==============================] - 0s 4ms/step - loss: 8306896142336.0000 -
mae: 2087316.3750 - val_loss: 10836413251584.0000 - val_mae: 2581414.5000
Epoch 47/100
12/12 [==============================] - 0s 4ms/step - loss: 8298181951488.0000 -
mae: 2086169.6250 - val_loss: 10824488845312.0000 - val_mae: 2579883.5000
Epoch 48/100
12/12 [==============================] - 0s 4ms/step - loss: 8289478770688.0000 -
mae: 2084951.2500 - val_loss: 10811804221440.0000 - val_mae: 2578255.2500
Epoch 49/100
12/12 [==============================] - 0s 4ms/step - loss: 8280316313600.0000 -
mae: 2083702.3750 - val_loss: 10798575386624.0000 - val_mae: 2576555.7500
Epoch 50/100
12/12 [==============================] - 0s 4ms/step - loss: 8270556692480.0000 -
mae: 2082406.6250 - val_loss: 10785101185024.0000 - val_mae: 2574825.7500
Epoch 51/100
12/12 [==============================] - 0s 4ms/step - loss: 8261131567104.0000 -
mae: 2081051.6250 - val_loss: 10770180997120.0000 - val_mae: 2572911.0000
Epoch 52/100
12/12 [==============================] - 0s 4ms/step - loss: 8250162937856.0000 -
mae: 2079618.6250 - val_loss: 10756046192640.0000 - val_mae: 2571088.5000
Epoch 53/100
12/12 [==============================] - 0s 4ms/step - loss: 8239305457664.0000 -
mae: 2078183.5000 - val_loss: 10741542289408.0000 - val_mae: 2569221.2500
Epoch 54/100
12/12 [==============================] - 0s 4ms/step - loss: 8228851154944.0000 -
mae: 2076679.5000 - val_loss: 10724985274368.0000 - val_mae: 2567097.0000
Epoch 55/100
12/12 [==============================] - 0s 4ms/step - loss: 8216875892736.0000 -
mae: 2075104.3750 - val_loss: 10708903264256.0000 - val_mae: 2565025.5000
Epoch 56/100
12/12 [==============================] - 0s 3ms/step - loss: 8205207339008.0000 -
mae: 2073479.8750 - val_loss: 10692434329600.0000 - val_mae: 2562902.2500
Epoch 57/100
12/12 [==============================] - 0s 4ms/step - loss: 8193299709952.0000 -
mae: 2071868.0000 - val_loss: 10675296403456.0000 - val_mae: 2560696.2500
Epoch 58/100
12/12 [==============================] - 0s 3ms/step - loss: 8180528578560.0000 -
mae: 2070130.2500 - val_loss: 10657480048640.0000 - val_mae: 2558400.0000
Epoch 59/100
12/12 [==============================] - 0s 4ms/step - loss: 8167743815680.0000 -
mae: 2068397.3750 - val_loss: 10638615117824.0000 - val_mae: 2555974.2500
```

```
Epoch 60/100
12/12 [==============================] - 0s 4ms/step - loss: 8154196738048.0000 -
mae: 2066547.3750 - val_loss: 10620355215360.0000 - val_mae: 2553613.0000
Epoch 61/100
12/12 [==============================] - 0s 4ms/step - loss: 8141112606720.0000 -
mae: 2064716.6250 - val_loss: 10601271132160.0000 - val_mae: 2551141.2500
Epoch 62/100
12/12 [==============================] - 0s 4ms/step - loss: 8127658328064.0000 -
mae: 2062905.6250 - val_loss: 10581428928512.0000 - val_mae: 2548571.0000
Epoch 63/100
12/12 [==============================] - 0s 4ms/step - loss: 8112932126720.0000 -
mae: 2060875.3750 - val_loss: 10561908637696.0000 - val_mae: 2546031.7500
Epoch 64/100
12/12 [==============================] - 0s 4ms/step - loss: 8098348007424.0000 -
mae: 2058916.8750 - val_loss: 10540915097600.0000 - val_mae: 2543311.7500
Epoch 65/100
12/12 [==============================] - 0s 3ms/step - loss: 8083379060736.0000 -
mae: 2056800.5000 - val_loss: 10519066968064.0000 - val_mae: 2540474.0000
Epoch 66/100
12/12 [==============================] - 0s 4ms/step - loss: 8067591700480.0000 -
mae: 2054681.6250 - val_loss: 10497942355968.0000 - val_mae: 2537718.2500
Epoch 67/100
12/12 [==============================] - 0s 3ms/step - loss: 8052057571328.0000 -
mae: 2052540.1250 - val_loss: 10475594055680.0000 - val_mae: 2534802.7500
Epoch 68/100
12/12 [==============================] - 0s 4ms/step - loss: 8035758505984.0000 -
mae: 2050275.3750 - val_loss: 10451897286656.0000 - val_mae: 2531706.0000
Epoch 69/100
12/12 [==============================] - 0s 3ms/step - loss: 8018593316864.0000 -
mae: 2047933.5000 - val_loss: 10428453224448.0000 - val_mae: 2528638.0000
Epoch 70/100
12/12 [==============================] - 0s 4ms/step - loss: 8001782022144.0000 -
mae: 2045618.6250 - val_loss: 10403681665024.0000 - val_mae: 2525398.5000
Epoch 71/100
12/12 [==============================] - 0s 4ms/step - loss: 7983371649024.0000 -
mae: 2043119.1250 - val_loss: 10379485773824.0000 - val_mae: 2522223.5000
Epoch 72/100
12/12 [==============================] - 0s 4ms/step - loss: 7965872488448.0000 -
mae: 2040688.5000 - val_loss: 10353930928128.0000 - val_mae: 2518880.0000
Epoch 73/100
12/12 [==============================] - 0s 4ms/step - loss: 7947195252736.0000 -
mae: 2038120.1250 - val_loss: 10328334139392.0000 - val_mae: 2515510.2500
Epoch 74/100
12/12 [==============================] - 0s 4ms/step - loss: 7928993546240.0000 -
mae: 2035534.6250 - val_loss: 10300686336000.0000 - val_mae: 2511881.7500
Epoch 75/100
12/12 [==============================] - 0s 4ms/step - loss: 7908526915584.0000 -
mae: 2032749.3750 - val_loss: 10273522974720.0000 - val_mae: 2508297.7500
Epoch 76/100
12/12 [==============================] - 0s 4ms/step - loss: 7888500686848.0000 -
mae: 2030017.7500 - val_loss: 10246913261568.0000 - val_mae: 2504775.0000
Epoch 77/100
12/12 [==============================] - 0s 4ms/step - loss: 7869286580224.0000 -
mae: 2027309.3750 - val_loss: 10217249046528.0000 - val_mae: 2500856.0000
Epoch 78/100
12/12 [==============================] - 0s 4ms/step - loss: 7847641874432.0000 -
mae: 2024288.3750 - val_loss: 10188230754304.0000 - val_mae: 2497006.0000
Epoch 79/100
12/12 [==============================] - 0s 4ms/step - loss: 7827384434688.0000 -
mae: 2021424.3750 - val_loss: 10157641695232.0000 - val_mae: 2492955.0000
Epoch 80/100
12/12 [==============================] - 0s 4ms/step - loss: 7805437214720.0000 -
mae: 2018359.8750 - val_loss: 10127257108480.0000 - val_mae: 2488908.7500
Epoch 81/100
```

```
12/12 [==============================] - 0s 5ms/step - loss: 7782753370112.0000 -
mae: 2015268.0000 - val_loss: 10096546414592.0000 - val_mae: 2484819.7500
Epoch 82/100
12/12 [==============================] - 0s 4ms/step - loss: 7760837083136.0000 -
mae: 2012157.8750 - val_loss: 10064101376000.0000 - val_mae: 2480486.2500
Epoch 83/100
12/12 [==============================] - 0s 4ms/step - loss: 7737548210176.0000 -
mae: 2008900.8750 - val_loss: 10031988736000.0000 - val_mae: 2476185.5000
Epoch 84/100
12/12 [==============================] - 0s 4ms/step - loss: 7714044379136.0000 -
mae: 2005615.1250 - val_loss: 9998680719360.0000 - val_mae: 2471725.0000
Epoch 85/100
12/12 [==============================] - 0s 4ms/step - loss: 7690293608448.0000 -
mae: 2002207.8750 - val_loss: 9964921815040.0000 - val_mae: 2467183.0000
Epoch 86/100
12/12 [==============================] - 0s 4ms/step - loss: 7665430822912.0000 -
mae: 1998725.6250 - val_loss: 9931540398080.0000 - val_mae: 2462687.0000
Epoch 87/100
12/12 [==============================] - 0s 4ms/step - loss: 7641321439232.0000 -
mae: 1995392.5000 - val_loss: 9897037004800.0000 - val_mae: 2458019.2500
Epoch 88/100
12/12 [==============================] - 0s 5ms/step - loss: 7616945192960.0000 -
mae: 1991842.6250 - val_loss: 9861790171136.0000 - val_mae: 2453239.5000
Epoch 89/100
12/12 [==============================] - 0s 4ms/step - loss: 7591546585088.0000 -
mae: 1988189.7500 - val_loss: 9824216547328.0000 - val_mae: 2448172.2500
Epoch 90/100
12/12 [==============================] - 0s 4ms/step - loss: 7564677873664.0000 -
mae: 1984394.2500 - val_loss: 9787857174528.0000 - val_mae: 2443225.0000
Epoch 91/100
12/12 [==============================] - 0s 4ms/step - loss: 7538702024704.0000 -
mae: 1980576.8750 - val_loss: 9751126605824.0000 - val_mae: 2438225.0000
Epoch 92/100
12/12 [==============================] - 0s 4ms/step - loss: 7511809196032.0000 -
mae: 1976901.5000 - val_loss: 9714061541376.0000 - val_mae: 2433147.5000
Epoch 93/100
12/12 [==============================] - 0s 4ms/step - loss: 7485333176320.0000 -
mae: 1973026.5000 - val_loss: 9676388302848.0000 - val_mae: 2427976.5000
Epoch 94/100
12/12 [==============================] - 0s 4ms/step - loss: 7457260175360.0000 -
mae: 1969080.6250 - val_loss: 9639493107712.0000 - val_mae: 2422897.2500
Epoch 95/100
12/12 [==============================] - 0s 4ms/step - loss: 7430101008384.0000 -
mae: 1965179.7500 - val_loss: 9601753808896.0000 - val_mae: 2417681.0000
Epoch 96/100
12/12 [==============================] - 0s 4ms/step - loss: 7403629707264.0000 -
mae: 1961308.8750 - val_loss: 9561919455232.0000 - val_mae: 2412172.7500
Epoch 97/100
12/12 [==============================] - 0s 4ms/step - loss: 7374567374848.0000 -
mae: 1957107.5000 - val_loss: 9522639798272.0000 - val_mae: 2406738.0000
Epoch 98/100
12/12 [==============================] - 0s 4ms/step - loss: 7345934434304.0000 -
mae: 1952977.6250 - val_loss: 9480762818560.0000 - val_mae: 2400927.7500
Epoch 99/100
12/12 [==============================] - 0s 4ms/step - loss: 7315983958016.0000 -
mae: 1948648.5000 - val_loss: 9439486672896.0000 - val_mae: 2395161.7500
Epoch 100/100
12/12 [==============================] - 0s 4ms/step - loss: 7286252634112.0000 -
mae: 1944272.7500 - val_loss: 9397463941120.0000 - val_mae: 2389289.7500
```

In [201…

```
#Predict on test data
predictions_ann = model_ann.predict(test_features)
print("Predicted values are: ", predictions_ann)
print("Real values are: ", test_labels)
```

```
5/5 [==============================] - 0s 1ms/step
Predicted values are:  [[ 42615.09  ]
 [  8889.014 ]
 [ 24488.758 ]
 [211759.92  ]
 [549108.8   ]
 [ 37516.13  ]
 [ 25127.02  ]
 [804744.    ]
 [ 13933.417 ]
 [ 39720.35  ]
 [ 29416.95  ]
 [267578.16  ]
 [ 18471.438 ]
 [ 25095.633 ]
 [ 30999.34  ]
 [ 15272.85  ]
 [411308.72  ]
 [ 14915.089 ]
 [363114.6   ]
 [408331.66  ]
 [561129.56  ]
 [ 39777.145 ]
 [ 16016.005 ]
 [429819.1   ]
 [ 27601.309 ]
 [136792.45  ]
 [183220.61  ]
 [ 79229.625 ]
 [ 28532.137 ]
 [ 20606.89  ]
 [ 10256.864 ]
 [457213.78  ]
 [ 38969.273 ]
 [ 42456.965 ]
 [263359.03  ]
 [413625.03  ]
 [415142.7   ]
 [ 85397.34  ]
 [266223.56  ]
 [ 10658.734 ]
 [ 36780.344 ]
 [  4369.8745]
 [170219.14  ]
 [129031.78  ]
 [ 10961.356 ]
 [ 54141.43  ]
 [166766.73  ]
 [513559.25  ]
 [ 32046.09  ]
 [488425.3   ]
 [237283.36  ]
 [280005.84  ]
 [ 42462.24  ]
 [ 22162.303 ]
 [ 34551.223 ]
 [ 11054.315 ]
 [ 45088.105 ]
 [155386.8   ]
 [135939.14  ]
 [722852.3   ]
 [537864.2   ]
 [ 15532.376 ]
 [577710.56  ]
```

```
[ 36855.652 ]
[ 26340.105 ]
[199608.11  ]
[ 69807.125 ]
[480127.53  ]
[112902.    ]
[ 22053.984 ]
[ 39894.7   ]
[ 49394.715 ]
[ 26261.605 ]
[ 56310.195 ]
[ 38866.24  ]
[187147.67  ]
[ 23414.111 ]
[ 59355.06  ]
[285279.56  ]
[392897.8   ]
[ 33040.348 ]
[ 39345.758 ]
[ 37577.312 ]
[ 37458.566 ]
[ 45243.48  ]
[ 18782.492 ]
[ 51998.95  ]
[266927.78  ]
[ 16860.072 ]
[ 31694.713 ]
[506815.9   ]
[407488.6   ]
[ 24702.832 ]
[ 34333.2   ]
[  5155.5566]
[ 55025.332 ]
[ 41147.156 ]
[ 73005.78  ]
[ 23739.396 ]
[520318.16  ]
[210565.78  ]
[ 62931.81  ]
[ 24812.383 ]
[344603.9   ]
[ 15188.531 ]
[202489.89  ]
[ 27551.611 ]
[525878.8   ]
[ 68771.    ]
[  7940.405 ]
[ 11309.017 ]
[413868.16  ]
[752238.06  ]
[ 24491.498 ]
[102416.96  ]
[100287.09  ]
[387316.03  ]
[807171.7   ]
[ 65186.45  ]
[ 73309.27  ]
[497311.66  ]
[ 97023.    ]
[ 26911.736 ]
[ 20808.31  ]
[ 21661.871 ]
[446985.44  ]
[ 58183.895 ]
```

```
 [246734.95  ]
 [240863.86  ]
 [264001.28  ]
 [ 36283.973 ]
 [  4931.3335]
 [296499.16  ]
 [424246.6   ]
 [581836.2   ]
 [ 28858.912 ]
 [ 44182.82  ]
 [171607.67  ]
 [108044.2   ]
 [261980.    ]
 [274578.16  ]
 [ 35409.414 ]
 [ 21259.916 ]
 [ 39440.258 ]
 [  4183.0947]
 [ 27248.9   ]
 [  4660.4043]
 [704708.44  ]
 [463663.03  ]
 [ 11886.303 ]
 [ 14013.352 ]
 [ 22234.863 ]]
Real values are:  [1025000.  667500.  667500. 3900000. 2500000.  600000.  800000.
4000000.
  792500.  925000.  650000. 3250000.  925000.  775000.  600000.  817500.
 6000000.  692500. 2500000. 5000000. 7636250.  900000.  600000. 5000000.
  700000.  800000. 4000000.  840000.  832500.  575000.  667500. 5750000.
 1000000.  615000. 1700000.  925000. 3700000.  792500. 3100000.  900000.
  650000.  600000.  600000.  850000.  600000.  717500. 6000000. 3400000.
  575000. 4500000. 5000000. 2950000. 2725000.  925000. 1000000.  874125.
  575000. 5000000. 2900000. 7636250. 3250000.  667500. 5000000.  750000.
  700000. 1500000. 1000000. 6000000.  875000.  842500. 1300000. 1850000.
  950000.  575000.  900000.  832500.  650000. 1000000. 3500000. 3275000.
  800000.  925000.  575000.  635000.  575000.  925000. 4500000. 2500000.
  900000.  600000. 4250000. 4500000.  600000.  767500. 3500000.  832500.
  832500.  600000.  600000. 7636250. 5500000. 3250000.  950000. 4000000.
 2600000.  625000.  650000. 5000000. 4000000. 4000000. 1300000. 3500000.
 7636250.  925000.  667500. 4875000.  925000. 6000000.  575000. 1050000.
 2000000.  900000.  925000.  832500.  925000. 4250000.  575000. 2000000.
 4500000. 4000000.  625000.  575000. 4250000. 4500000. 5750000.  640000.
  832500. 7000000.  825000. 3550000. 6750000.  680000.  600000.  800000.
  715000. 4000000.  750000. 6500000. 7500000.  725000.  925000.  575000.]
```

In [207…

```
#Comparison with other models..
#Neural network - from the current code
mse_neural, mae_neural = model_ann.evaluate(test_features, test_labels)
print('Mean squared error from neural net: ', mse_neural)
print('Mean absolute error from neural net: ', mae_neural)
```

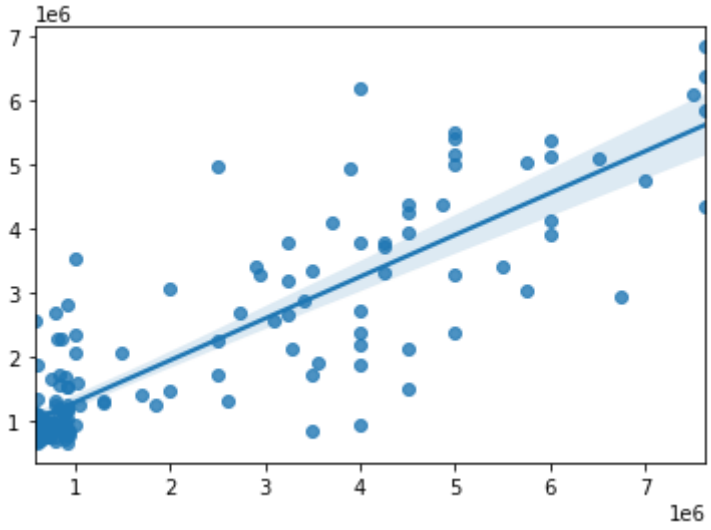
```
5/5 [==============================] - 0s 2ms/step - loss: 7742888083456.0000 - ma
e: 2047853.5000
Mean squared error from neural net:  7742888083456.0
Mean absolute error from neural net:  2047853.5
```

In [203…

```
sns.regplot(test_labels, predictions)
```

Out[203]:

```
<AxesSubplot:>
```



Decision Tree

In [157…

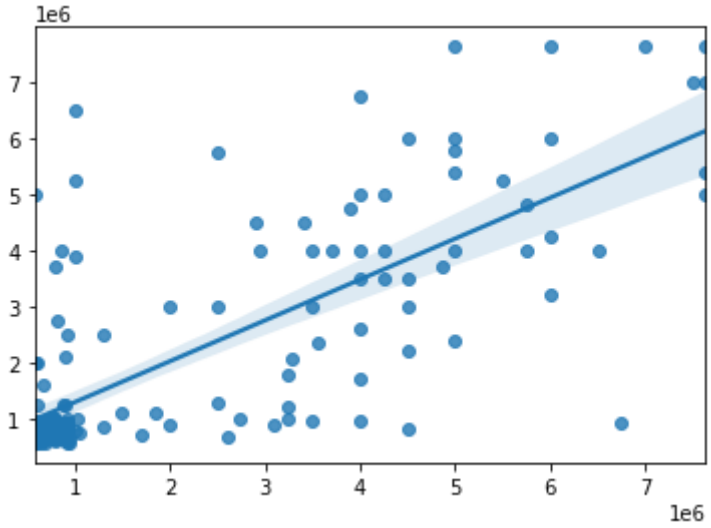
```
### Decision tree
tree = DecisionTreeRegressor()
tree.fit(train_features, train_labels)
y_pred_tree = tree.predict(test_features)
mse_dt = mean_squared_error(test_labels, y_pred_tree)
mae_dt = mean_absolute_error(test_labels, y_pred_tree)
print('Mean squared error using decision tree: ', mse_dt)
print('Mean absolute error using decision tree: ', mae_dt)
# R-Squared
r_square=r2_score(test_labels,y_pred_tree)
print("R2=",r2_score(test_labels,y_pred_tree))
#Adjusted r square
adjusted_r_square=1-(1-r_square)*(len(test_labels)-1)/(len(test_labels)-test_featu
print("Adjusted_R2=",adjusted_r_square)
y_pred_tree
```

```
Mean squared error using decision tree:  2009784833573.1907
Mean absolute error using decision tree:  869267.2697368421
R2= 0.5130524893551883
Adjusted_R2= 0.45934504332818704
```

Out[157]:
```
array([1000000.,  792500.,  925000., 4750000., 3000000.,  575000.,
        660000., 6750000.,  700000.,  575000.,  575000., 1000000.,
       2500000.,  874125.,  600000.,  832500., 4250000.,  575000.,
       1300000., 7636250., 7000000., 1250000.,  735000., 6000000.,
        775000.,  620000., 2600000.,  925000.,  832500.,  632500.,
        792500., 4838000., 3900000.,  640000.,  700000.,  925000.,
       4000000., 1000000.,  900000.,  832500.,  595000.,  874125.,
        792500., 4000000., 1250000.,  925000., 7636250., 4500000.,
        625000.,  832500., 4000000., 4000000., 1000000.,  832500.,
        792500., 1250000.,  600000., 2400000., 4500000., 7636250.,
       1800000., 1600000., 5800000.,  800000.,  650000., 1100000.,
       5250000., 3200000.,  925000.,  925000., 2500000., 1100000.,
        632500.,  800000.,  600000.,  925000.,  775000., 6500000.,
       4000000., 2075000., 3700000.,  925000.,  650000.,  925000.,
        625000., 1000000., 6000000., 5750000.,  725000.,  950000.,
       4000000., 3000000.,  892500., 1050000.,  950000.,  925000.,
        742500., 2000000.,  742500., 5400000., 5250000., 1200000.,
        575000., 4000000.,  675000.,  832500.,  874125., 5400000.,
       5000000.,  950000.,  874125., 3000000., 5000000.,  925000.,
        640000., 3700000.,  925000., 6000000.,  725000.,  750000.,
       3000000., 2100000.,  925000.,  925000.,  925000., 5000000.,
       5000000.,  900000., 2200000., 1700000.,  680000.,  575000.,
       3500000., 3500000., 4000000.,  925000.,  925000., 7636250.,
       2750000., 2350000.,  925000.,  925000.,  667500.,  650000.,
        950000., 3500000., 1000000., 4000000., 7000000.,  667500.,
        832500., 2000000.])
```

In [158…
```
sns.regplot(test_labels, y_pred_tree)
```

Out[158]:
```
<AxesSubplot:>
```



random forest

In [159…
```
# Import the model we are using
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 1000,max_depth=9, random_state = 42)
# Train the model on training data
rf.fit(train_features, train_labels)
```

Out[159]:
```
RandomForestRegressor(max_depth=9, n_estimators=1000, random_state=42)
```

In [160…
```
# Use the forest's predict method on the test data
predictions = rf.predict(test_features)
```

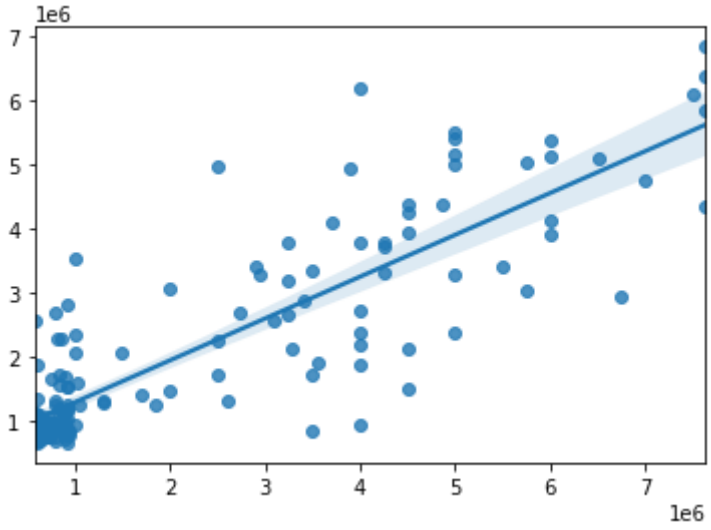
```
mse_rf = mean_squared_error(test_labels, predictions)
mae_rf = mean_absolute_error(test_labels, predictions)
print('Mean squared error using random forest: ', mse_dt)
print('Mean absolute error using random forest: ', mae_dt)
# R-Squared
r_square=r2_score(test_labels,predictions)
print("R2=",r2_score(test_labels,predictions))
#Adjusted r square
adjusted_r_square=1-(1-r_square)*(len(test_labels)-1)/(len(test_labels)-test_featu
print("Adjusted_R2=",adjusted_r_square)
predictions
```

```
Mean squared error using random forest:  2009784833573.1907
Mean absolute error using random forest:  869267.2697368421
R2= 0.7174681283557844
Adjusted_R2= 0.6863065248656136
```

Out[160]:
```
array([1588649.1713323 , 1018099.16284804,  937074.36845909,
       4924618.67434405, 1723663.99132   ,  695419.17140914,
        752524.18811569, 6187236.48600859,  976641.12720206,
        657937.46356409,  704002.22275323, 3793608.38103197,
       1532528.64027613,  855555.86511742,  670841.52016495,
        873093.54387783, 4113047.51068188,  778321.62571403,
       2261963.2504155 , 5508849.9053564 , 6835412.18465501,
       1009610.97712916,  946784.86047694, 5000583.05152935,
        739716.84295654, 1317413.34259049, 2703286.2856703 ,
       1196931.4206431 , 1554783.57565748,  693678.06197149,
       1007121.44354173, 5028563.37468593, 2342916.84759351,
       1328909.26910421, 1404868.19211874, 1530261.33794629,
       4106478.7139354 , 1238168.2548152 , 2571663.97385864,
        922066.80880673,  728741.99589948,  833383.28167875,
        910153.2795972 , 2293096.90923359,  877532.13185036,
       1058980.45420827, 5135306.15234295, 2872322.94321658,
        960486.94843998, 2111450.93171276, 3280813.76474288,
       3289205.47441768, 2701935.61828673,  840601.74901634,
        950852.4807833 ,  835487.95843129, 1096183.13199929,
       2386290.23026936, 3414083.63166366, 6380658.91932529,
       3195670.16887869,  821307.01745642, 5162202.50406344,
       1651230.05021512,  772464.02982927, 2049993.36840077,
       3529307.66733964, 3900843.19230839, 1131928.30524475,
        911282.33021321, 1324737.86295787, 1242956.73993307,
        850367.54951034,  910049.58865024,  707578.47906044,
       1120381.58374823,  734191.80328905, 2074369.8520024 ,
       3329816.60500877, 2120362.32991257, 2684399.01304623,
       1173110.31033554,  704770.41642201,  840852.6659668 ,
       1125899.49119004, 2799039.01128673, 4372737.10829451,
       4962774.42439409,  904293.61414802,  957187.10549433,
       3767864.26086002, 4253944.09961718,  726565.89357062,
        900699.68249013,  852754.48982035, 1727841.42007584,
        872896.42994073, 1860195.92206326,  840946.96064072,
       5847301.61041799, 3415062.25760977, 2647748.72970846,
        785260.27554225, 3770195.13277569, 1301625.30158459,
       1077314.80706317,  848084.26055714, 5398224.51712898,
       2379380.81775625,  943172.22566178, 1268708.91754641,
       1718513.04688375, 4338075.06934012, 1263608.64387367,
        932291.33825316, 4381357.7647061 , 1139888.18887619,
       5358727.13929765, 1087928.02730863, 1239663.51607862,
       1476673.34167638, 1702025.25032083,  795853.00094601,
        887750.95539843, 1032414.88679426, 3298197.3110097 ,
       2554299.71353308, 3066328.90908352, 1492085.05342917,
       1861763.03940891,  695326.14171997,  880480.33179811,
       3728204.94293064, 3945725.26468227, 3025836.63267132,
        892587.78426396,  921927.58032283, 4762108.59159033,
       2277717.58133883, 1899553.76871368, 2921284.6732785 ,
        817086.88512181,  779641.18331063,  683682.91422556,
        745959.82163192, 2171297.44446059,  968881.40258104,
       5096066.98482085, 6096890.35283511,  761924.95030313,
        877027.35681613, 1114992.70578925])
```

In [161…
```
sns.regplot(test_labels, predictions)
```

Out[161]:
```
<AxesSubplot:>
```



Hyperparameter tuning in random forest

In [162…
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 1000,max_depth=9,random_state = 42)
from pprint import pprint
# Look at parameters used by our current forest
print('Parameters currently in use:\n')
pprint(rf.get_params())
```

```
Parameters currently in use:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': 9,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 1000,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 42,
 'verbose': 0,
 'warm_start': False}
```

In [163…
```
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(3,30, num =10)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
```

```
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [3, 6, 9, 12, 15, 18, 21, 24, 27, 30, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```

In [164…

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, r
# Fit the random search model
rf_random.fit(train_features, train_labels)
```

Out[164]:

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [3, 6, 9, 12, 15, 18, 21,
                                                      24, 27, 30, None],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [200, 400, 600, 800,
                                                         1000, 1200, 1400, 1600,
                                                         1800, 2000]},
                   random_state=42, verbose=2)
```

In [165…

```
rf_random.best_params_
```

Out[165]:

```
{'n_estimators': 200,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 30,
 'bootstrap': False}
```

In [166…

```
new=rf_random.predict(test_features)
new
```

Out[166]:
```
array([1360954.73958333, 1170402.5       ,  823322.96875   ,
       4540979.58333333, 2021573.95833333,  711691.09375   ,
        897473.75      , 5961441.14583333, 1089191.45833333,
        630356.25      ,  714328.64583333, 3976812.39583333,
       1226520.9375    ,  809208.95833333,  628873.95833333,
        966975.57291667, 4383577.29166667,  811968.90625   ,
       2743016.875     , 4803561.5625    , 6098422.60416667,
        994937.8125    , 1015978.02083333, 4747847.39583333,
        682440.10416667, 1774137.29166667, 2808376.45833333,
       1166812.29166667, 1278913.85416667,  663650.20833333,
       1223371.45833333, 4581335.83333333, 1921123.95833333,
       1440159.375     , 1766618.75      , 2496409.375     ,
       3738193.02083333, 1586859.79166667, 2622647.8125    ,
        973536.25      ,  787057.60416667,  885045.20833333,
       1223852.5       , 2388523.95833333,  904744.375     ,
       1456535.41666667, 4481638.95833333, 3014058.54166667,
        746494.84375   , 2606486.45833333, 3000464.0625    ,
       3244718.125     , 1717587.5       ,  832332.70833333,
       1056400.        , 1106820.20833333,  952627.65625   ,
       2364800.        , 3343838.85416667, 6170699.0625    ,
       2882235.20833333,  854876.66666667, 4817312.60416667,
       1617456.66666667,  706134.32291667, 2324731.66666667,
       3357199.79166667, 3745818.75      , 1214792.1875    ,
        929215.41666667, 1095992.08333333, 1470165.41666667,
        821306.45833333, 1114185.88541667,  742482.29166667,
       1558100.83333333,  735564.58333333, 1500263.54166667,
       3329916.77083333, 2540546.45833333, 2210979.27083333,
       1278285.41666667,  644175.        ,  794559.58333333,
        992262.44791667, 2197052.08333333, 3304388.54166667,
       4214183.125     ,  940813.28125   , 1077352.5       ,
       2920760.41666667, 4068497.08333333,  789489.375     ,
        987545.46875   , 1086058.28125   , 2162958.4375    ,
        912279.79166667, 1740913.95833333,  905917.44791667,
       5303914.6875    , 3100078.33333333, 2280426.04166667,
        887265.57291667, 4159156.5625    , 1635084.89583333,
       1160101.66666667,  832934.58333333, 4922040.72916667,
       2567224.6875    , 1135016.45833333, 1577551.97916667,
       2418033.54166667, 4600533.54166667, 1383749.58333333,
       1308733.33333333, 3703575.        , 1398672.70833333,
       5638485.        , 1217991.875     ,  956674.375     ,
       1779848.22916667, 2086919.0625    ,  844519.53125   ,
        892904.6875    , 1187222.5       , 3287685.52083333,
       1984685.20833333, 3247792.70833333, 1938436.77083333,
       2376298.85416667,  712096.25      , 1187754.79166667,
       3477628.22916667, 3964365.83333333, 3486466.66666667,
        885147.03125   , 1009697.86458333, 4178873.22916667,
       2028585.        , 2643339.79166667, 3256064.89583333,
        925464.58333333,  886916.14583333,  668899.27083333,
        865901.875     , 1736754.58333333, 1092307.44791667,
       4960520.10416667, 5299867.5       ,  925956.875     ,
        852658.4375    , 1333958.64583333])
```

Xgboost

In [195…
```
# evaluate an xgboost regression model on the housing dataset
from numpy import absolute
from pandas import read_csv
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedKFold
from xgboost import XGBRegressor

model_xg = XGBRegressor()
```

```
# define model evaluation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate model
scores = cross_val_score(model_xg, train_features, train_labels, scoring='neg_mean_
# force scores to be positive
scores = absolute(scores)
print('MAE: %.3f (%.3f)' % (scores.mean(), scores.std()) )
```

```
MAE: 811455.451 (118677.283)
```

In [169…

```
# fit model
model_xg.fit(train_features, train_labels)
y_test_pred_xg=model_xg.predict(test_features)
mse = mean_squared_error(y_test_pred_xg, test_labels)
print('mse',mse)
# R-Squared
r_square=r2_score(test_labels,y_test_pred_xg)
print("R2=",r2_score(test_labels,y_test_pred_xg))
#Adjusted r square
adjusted_r_square=1-(1-r_square)*(len(test_labels)-1)/(len(test_labels)-test_featu
print("Adjusted_R2=",adjusted_r_square)
```

```
mse 1396120190364.6282
R2= 0.661736301367975
Adjusted_R2= 0.6244278051953251
```

In [170…

```
y_test_pred_xg
```

Out[170]:

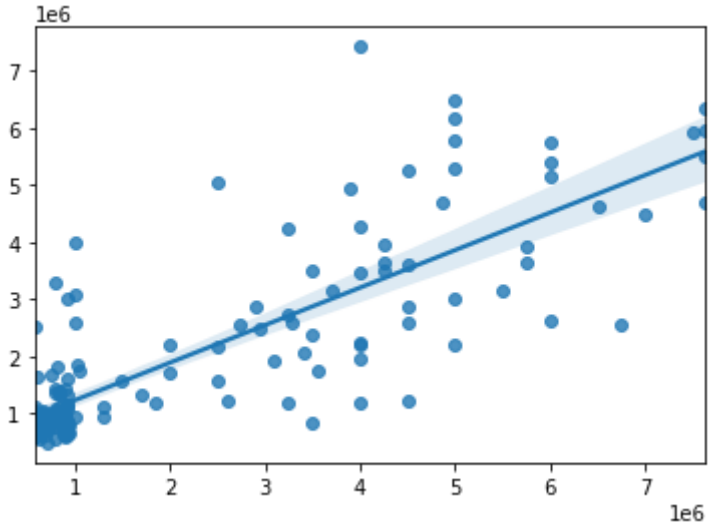
```
array([1834056.6 , 1038876.2 ,  933142.56, 4931075.5 , 1579747.9 ,
        645119.2 ,  857590.56, 7430079.  , 1352688.6 ,  641402.  ,
        658220.  , 2727285.5 , 1612770.9 ,  752122.4 ,  624416.44,
        992735.56, 5397799.5 ,  613522.5 , 2167147.5 , 6188096.  ,
       6338215.5 ,  971161.2 ,  974195.7 , 5285922.  ,  732327.4 ,
       1413123.6 , 2214463.2 , 1367846.1 , 1384599.4 ,  684068.3 ,
        955971.9 , 3908145.  , 3092920.  ,  937291.4 , 1328192.5 ,
       1066467.1 , 3135059.  , 1091743.5 , 1913035.2 ,  903759.  ,
        821692.2 ,  838105.3 ,  899910.94,  996732.5 ,  737622.6 ,
        479442.22, 5159082.  , 2046927.4 ,  743782.7 , 2872763.  ,
       2201771.8 , 2488728.5 , 2557586.5 ,  805653.8 ,  931622.9 ,
        612114.06,  860896.9 , 3015151.2 , 2876496.2 , 5944836.5 ,
       1193201.9 , 1003638.  , 5799798.5 , 1673631.5 ,  831420.8 ,
       1574151.8 , 4002807.5 , 2627146.2 ,  759030.4 ,  751166.5 ,
       1094579.9 , 1168297.  ,  642755.25, 1031058.3 ,  585778.3 ,
       1311747.  ,  725162.5 , 2582392.2 , 3485834.  , 2596648.5 ,
       3277333.  ,  813875.3 ,  572159.3 ,  801423.8 ,  734715.06,
       3005254.8 , 2590340.2 , 5053705.5 ,  697972.94,  599936.4 ,
       3647209.2 , 5272832.5 ,  774749.44,  915807.9 ,  816187.94,
       1110236.1 ,  880521.4 , 1622394.5 ,  809892.4 , 5505436.5 ,
       3135875.5 , 4221973.5 ,  794281.4 , 4285852.  , 1208966.4 ,
        752499.8 ,  788673.25, 6504809.5 , 1945963.8 , 1182348.9 ,
        930602.4 , 2359976.  , 4704893.5 , 1333037.2 ,  893628.3 ,
       4697802.  ,  626638.3 , 5764372.5 ,  598720.4 , 1753955.9 ,
       1695935.2 , 1408596.6 , 1135113.9 ,  869225.1 , 1130053.8 ,
       3972734.  , 2512968.8 , 2183303.8 , 1227713.  , 2233432.5 ,
        559116.94,  929919.44, 3512970.2 , 3610580.5 , 3638445.  ,
        919435.5 ,  861826.56, 4480818.5 , 1802402.1 , 1749039.6 ,
       2545817.8 ,  847301.25,  677938.56,  556489.6 ,  735227.1 ,
       3477922.  ,  985674.75, 4618962.5 , 5931602.  ,  790626.25,
        854423.8 , 1107441.6 ], dtype=float32)
```

In [172…

```
sns.regplot(test_labels, y_test_pred_xg)
```

Out[172]:

```
<AxesSubplot:>
```



Adaboost

In [178…
```
from sklearn.ensemble import AdaBoostRegressor
model_ada = AdaBoostRegressor(n_estimators=20, random_state=1)

# define model evaluation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate model
scores = cross_val_score(model_ada, train_features, train_labels, scoring='neg_mea

# fit model
model_ada.fit(train_features, train_labels)
y_test_pred_ada=model_ada.predict(test_features)
mae_ada = mean_absolute_error(test_labels, y_test_pred_ada)

print('MAE',mae_ada)
```

```
MAE 927844.6652242817
```

In [180…
```
# fit model
model_ada.fit(train_features, train_labels)
y_test_pred_ada=model_ada.predict(test_features)
mse = mean_squared_error(y_test_pred_ada, test_labels)
print('mse',mse)
# R-Squared
r_square=r2_score(test_labels,y_test_pred_ada)
print("R2=",r2_score(test_labels,y_test_pred_ada))
#Adjusted r square
adjusted_r_square=1-(1-r_square)*(len(test_labels)-1)/(len(test_labels)-test_featu
print("Adjusted_R2=",adjusted_r_square)
```

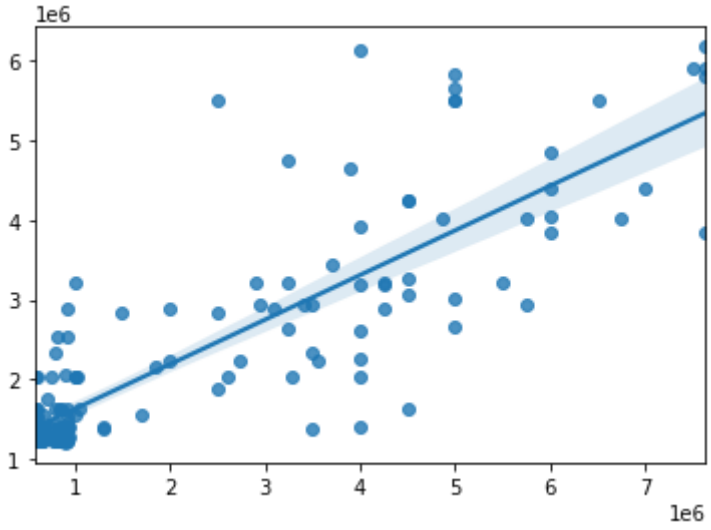
```
mse 1323840174917.4856
R2= 0.6792489091871805
Adjusted_R2= 0.6438719506416489
```

In [181…
```
y_test_pred_ada
```

Out[181]:
```
array([2039668.2464455 , 1370873.4939759 , 1234293.47826087,
       4638842.10526316, 1880626.98412698, 1234293.47826087,
       1270764.92537313, 6128247.76785714, 1234293.47826087,
       1232934.06593407, 1232934.06593407, 4740750.        ,
       1457083.33333333, 1234293.47826087, 1234293.47826087,
       1270764.92537313, 4047813.10679612, 1280846.15384615,
       2844380.95238095, 5511132.0754717 , 6177306.81818182,
       1234293.47826087, 1397316.        , 5658550.53191489,
       1280846.15384615, 1457083.33333333, 2610652.17391304,
       1397316.        , 1301062.5       , 1234293.47826087,
       1397316.        , 4026757.8125    , 2039668.2464455 ,
       1521166.66666667, 1565685.18518519, 2887743.24324324,
       3440943.39622641, 1627190.05847953, 2887743.24324324,
       1204877.20588235, 1234293.47826087, 1270764.92537313,
       1627190.05847953, 1627190.05847953, 1270764.92537313,
       1759863.01369863, 4409619.93243243, 2930080.64516129,
       1397316.        , 3067090.90909091, 3008164.73988439,
       2936430.76923077, 2240784.72222222, 1234293.47826087,
       1565685.18518519, 1370873.4939759 , 1627190.05847953,
       2672693.66197183, 3215521.84466019, 5796337.20930233,
       3208620.68965517, 1270764.92537313, 5511132.0754717 ,
       2037772.15189873, 1280846.15384615, 2844380.95238095,
       3215521.84466019, 3849038.46153846, 1370873.4939759 ,
       1280846.15384615, 1370873.4939759 , 2150906.59340659,
       1397316.        , 1234293.47826087, 1234293.47826087,
       1370873.4939759 , 1270764.92537313, 2037772.15189873,
       2936430.76923077, 2039668.2464455 , 2322869.64285714,
       1370873.4939759 , 1234293.47826087, 1270764.92537313,
       1627190.05847953, 2534120.37037037, 4250162.16216216,
       5511132.0754717 , 1370873.4939759 , 1370873.4939759 ,
       3215521.84466019, 4250000.        , 1280846.15384615,
       1270764.92537313, 1370873.4939759 , 1627391.30434783,
       1232934.06593407, 2037772.15189873, 1270764.92537313,
       5902710.38251366, 3215521.84466019, 2647413.79310345,
       1280846.15384615, 3929655.17241379, 2039668.2464455 ,
       1270764.92537313, 1234293.47826087, 5829238.0952381 ,
       3188564.67661692, 1397316.        , 1397316.        ,
       2322869.64285714, 3849038.46153846, 1521166.66666667,
       1566778.84615385, 4026757.8125    , 1627190.05847953,
       4854916.66666667, 1370873.4939759 , 1627190.05847953,
       2240784.72222222, 2068589.74358974, 1270764.92537313,
       1270764.92537313, 1397316.        , 2887743.24324324,
       2039668.2464455 , 2887743.24324324, 1627190.05847953,
       2039668.2464455 , 1232934.06593407, 1565685.18518519,
       3188564.67661692, 3255378.78787879, 2930080.64516129,
       1280846.15384615, 1232934.06593407, 4409619.93243243,
       2534120.37037037, 2240784.72222222, 4027678.57142857,
       1370873.4939759 , 1270764.92537313, 1232934.06593407,
       1280846.15384615, 2248958.33333333, 1270764.92537313,
       5511132.0754717 , 5902710.38251366, 1397316.        ,
       1234293.47826087, 1280846.15384615])
```

In [183…
```
sns.regplot(test_labels, y_test_pred_ada)
```

Out[183]:
```
<AxesSubplot:>
```



In [ ]: