

```
In [1]: import findspark
findspark.init()
findspark.find()

import pyspark
from pyspark import SparkContext, SparkConf, SQLContext
from pyspark.sql import SparkSession

from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .master("local[*]") \
    .appName("GenericAppName") \
    .getOrCreate()
sc=spark.sparkContext
sqlContext = SQLContext(sc)
```

C:\Spark\Spark\spark-3.3.0-bin-hadoop3\spark-3.3.0-bin-hadoop3\python\pyspark\sql\context.py:112: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.  
warnings.warn(

## Task I

```
In [2]: # Ingest data 2015-2022
from pyspark import SparkFiles

players_15 = spark.read.csv('FIFA_DATA/players_15.csv',header=True, inferSchema = True)
players_16 = spark.read.csv('FIFA_DATA/players_16.csv',header=True, inferSchema = True)
players_17 = spark.read.csv('FIFA_DATA/players_17.csv',header=True, inferSchema = True)
players_18 = spark.read.csv('FIFA_DATA/players_18.csv',header=True, inferSchema = True)
players_19 = spark.read.csv('FIFA_DATA/players_19.csv',header=True, inferSchema = True)
players_20 = spark.read.csv('FIFA_DATA/players_20.csv',header=True, inferSchema = True)
players_21 = spark.read.csv('FIFA_DATA/players_21.csv',header=True, inferSchema = True)
players_22 = spark.read.csv('FIFA_DATA/players_22.csv',header=True, inferSchema = True)
```

```
In [3]: print(players_15.columns[55])

movement_agility
```

```
In [4]: #Add new column for the year
from pyspark.sql.functions import lit
```

```
players_15 = players_15.withColumn('Year', lit(2015))
players_16 = players_16.withColumn('Year', lit(2016))
players_17 = players_17.withColumn('Year', lit(2017))
players_18 = players_18.withColumn('Year', lit(2018))
players_19 = players_19.withColumn('Year', lit(2019))
players_20 = players_20.withColumn('Year', lit(2020))
players_21 = players_21.withColumn('Year', lit(2021))
players_22 = players_22.withColumn('Year', lit(2022))
```

In [5]: *#Ensure every record can be uniquely identified  
#combine fifa id and year as index*

```
from pyspark.sql import functions as sf
```

```
players_15 = players_15.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col('Year'))))
players_16 = players_16.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col('Year'))))
players_17 = players_17.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col('Year'))))
players_18 = players_18.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col('Year'))))
players_19 = players_19.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col('Year'))))
players_20 = players_20.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col('Year'))))
players_21 = players_21.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col('Year'))))
players_22 = players_22.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_'), sf.col('Year'))))
```

In [7]: *# union the data to one dataset*

```
data = players_15.union(players_16)
data = data.union(players_17)
data = data.union(players_18)
data = data.union(players_19)
data = data.union(players_20)
data = data.union(players_21)
data = data.union(players_22)
```

In [8]: *#ingest the dataset into postgres*

```
db_properties={}
#update your db username
db_properties['username']="postgres"
#update your db password
db_properties['password']="20000823"
#make sure you got the right port number here
db_properties['url']="jdbc:postgresql://localhost:5432/postgres"
#make sure you had the Postgres JAR file in the right location
db_properties['driver']="org.postgresql.Driver"
db_properties['table']="fifa.data"
```

```
data.write.format("jdbc")\
.mode("overwrite")\
.option("url", db_properties['url'])\
.option("dbtable", db_properties['table'])\
.option("user", db_properties['username'])\
.option("password", db_properties['password'])\
.option("Driver", db_properties['driver'])\
.save()
```

In [9]: `data.printSchema()`

```
root
|-- sofiya_id: integer (nullable = true)
|-- player_url: string (nullable = true)
|-- short_name: string (nullable = true)
|-- long_name: string (nullable = true)
|-- player_positions: string (nullable = true)
|-- overall: integer (nullable = true)
|-- potential: integer (nullable = true)
|-- value_eur: double (nullable = true)
|-- wage_eur: double (nullable = true)
|-- age: integer (nullable = true)
|-- dob: timestamp (nullable = true)
|-- height_cm: integer (nullable = true)
|-- weight_kg: integer (nullable = true)
|-- club_team_id: double (nullable = true)
|-- club_name: string (nullable = true)
|-- league_name: string (nullable = true)
|-- league_level: integer (nullable = true)
|-- club_position: string (nullable = true)
|-- club_jersey_number: integer (nullable = true)
|-- club_loaned_from: string (nullable = true)
|-- club_joined: timestamp (nullable = true)
|-- club_contract_valid_until: integer (nullable = true)
|-- nationality_id: integer (nullable = true)
|-- nationality_name: string (nullable = true)
|-- nation_team_id: double (nullable = true)
|-- nation_position: string (nullable = true)
|-- nation_jersey_number: integer (nullable = true)
|-- preferred_foot: string (nullable = true)
|-- weak_foot: integer (nullable = true)
|-- skill_moves: integer (nullable = true)
|-- international_reputation: integer (nullable = true)
|-- work_rate: string (nullable = true)
|-- body_type: string (nullable = true)
|-- real_face: string (nullable = true)
|-- release_clause_eur: string (nullable = true)
|-- player_tags: string (nullable = true)
|-- player_traits: string (nullable = true)
|-- pace: integer (nullable = true)
|-- shooting: integer (nullable = true)
|-- passing: integer (nullable = true)
|-- dribbling: integer (nullable = true)
|-- defending: integer (nullable = true)
|-- physic: integer (nullable = true)
|-- attacking_crossing: integer (nullable = true)
```

```
|-- attacking_finishing: integer (nullable = true)
|-- attacking_heading_accuracy: integer (nullable = true)
|-- attacking_short_passing: integer (nullable = true)
|-- attacking_volleys: integer (nullable = true)
|-- skill_dribbling: integer (nullable = true)
|-- skill_curve: integer (nullable = true)
|-- skill_fk_accuracy: integer (nullable = true)
|-- skill_long_passing: integer (nullable = true)
|-- skill_ball_control: integer (nullable = true)
|-- movement_acceleration: integer (nullable = true)
|-- movement_sprint_speed: integer (nullable = true)
|-- movement_agility: integer (nullable = true)
|-- movement_reactions: integer (nullable = true)
|-- movement_balance: integer (nullable = true)
|-- power_shot_power: integer (nullable = true)
|-- power_jumping: integer (nullable = true)
|-- power_stamina: integer (nullable = true)
|-- power_strength: integer (nullable = true)
|-- power_long_shots: integer (nullable = true)
|-- mentality_aggression: integer (nullable = true)
|-- mentality_interceptions: integer (nullable = true)
|-- mentality_positioning: integer (nullable = true)
|-- mentality_vision: integer (nullable = true)
|-- mentality_penalties: integer (nullable = true)
|-- mentality_composure: string (nullable = true)
|-- defending_marking_awareness: integer (nullable = true)
|-- defending_standing_tackle: integer (nullable = true)
|-- defending_sliding_tackle: integer (nullable = true)
|-- goalkeeping_diving: integer (nullable = true)
|-- goalkeeping_handling: integer (nullable = true)
|-- goalkeeping_kicking: integer (nullable = true)
|-- goalkeeping_positioning: integer (nullable = true)
|-- goalkeeping_reflexes: integer (nullable = true)
|-- goalkeeping_speed: integer (nullable = true)
|-- ls: string (nullable = true)
|-- st: string (nullable = true)
|-- rs: string (nullable = true)
|-- lw: string (nullable = true)
|-- lf: string (nullable = true)
|-- cf: string (nullable = true)
|-- rf: string (nullable = true)
|-- rw: string (nullable = true)
|-- lam: string (nullable = true)
|-- cam: string (nullable = true)
|-- ram: string (nullable = true)
```

```
-- lm: string (nullable = true)
-- lcm: string (nullable = true)
-- cm: string (nullable = true)
-- rcm: string (nullable = true)
-- rm: string (nullable = true)
-- lwb: string (nullable = true)
-- ldm: string (nullable = true)
-- cdm: string (nullable = true)
-- rdm: string (nullable = true)
-- rwb: string (nullable = true)
-- lb: string (nullable = true)
-- lcb: string (nullable = true)
-- cb: string (nullable = true)
-- rcb: string (nullable = true)
-- rb: string (nullable = true)
-- gk: string (nullable = true)
-- player_face_url: string (nullable = true)
-- club_logo_url: string (nullable = true)
-- club_flag_url: string (nullable = true)
-- nation_logo_url: string (nullable = true)
-- nation_flag_url: string (nullable = true)
-- Year: integer (nullable = false)
-- data_ID: string (nullable = true)
```

```
In [10]: # read the data back from the postgres
data_read = sqlContext.read.format("jdbc")\
    .option("url", db_properties['url'])\
    .option("dbtable", db_properties['table'])\
    .option("user", db_properties['username'])\
    .option("password", db_properties['password'])\
    .option("Driver", db_properties['driver'])\
    .load()

data_read.show(1, vertical=True)
```

-RECORD 0-----	
sofifa_id	190745
player_url	<a href="https://sofifa.co...">https://sofifa.co...</a>
short_name	M. Silvestri
long_name	Marco Silvestri
player_positions	GK
overall	69
potential	74
value_eur	1100000.0
wage_eur	10000.0
age	25
dob	1991-03-02 00:00:00
height_cm	191
weight_kg	80
club_team_id	8.0
club_name	Leeds United
league_name	English League Ch...
league_level	2
club_position	RES
club_jersey_number	12
club_loaned_from	null
club_joined	2014-07-09 00:00:00
club_contract_valid_until	2018
nationality_id	27
nationality_name	Italy
nation_team_id	null
nation_position	null
nation_jersey_number	null
preferred_foot	Right
weak_foot	2
skill_moves	1
international_reputation	1
work_rate	Medium/Medium
body_type	Lean (185+)
real_face	No
release_clause_eur	null
player_tags	null
player_traits	Puncher
pace	null
shooting	null
passing	null
dribbling	null
defending	null
physic	null
attacking_crossing	12

attacking_finishing	20
attacking_heading_accuracy	12
attacking_short_passing	22
attacking_volleys	20
skill_dribbling	18
skill_curve	16
skill_fk_accuracy	11
skill_long_passing	20
skill_ball_control	22
movement_acceleration	58
movement_sprint_speed	61
movement_agility	65
movement_reactions	62
movement_balance	34
power_shot_power	16
power_jumping	74
power_stamina	34
power_strength	60
power_long_shots	16
mentality_aggression	31
mentality_interceptions	11
mentality_positioning	14
mentality_vision	49
mentality_penalties	16
mentality_composure	39
defending_marking_awareness	18
defending_standing_tackle	18
defending_sliding_tackle	16
goalkeeping_diving	78
goalkeeping_handling	59
goalkeeping_kicking	53
goalkeeping_positioning	67
goalkeeping_reflexes	76
goalkeeping_speed	60
ls	27+1
st	27+1
rs	27+1
lw	30
lf	29
cf	29
rf	29
rw	30
lam	31+1
cam	31+1
ram	31+1

lm	30+1
lcm	27+1
cm	27+1
rcm	27+1
rm	30+1
lwb	26+1
ldm	26+1
cdm	26+1
rdm	26+1
rwb	26+1
lb	26+1
lcb	27+1
cb	27+1
rcb	27+1
rb	26+1
gk	68+1
player_face_url	https://cdn.sofif...
club_logo_url	https://cdn.sofif...
club_flag_url	https://cdn.sofif...
nation_logo_url	null
nation_flag_url	https://cdn.sofif...
Year	2017
data_ID	190745_2017

only showing top 1 row

## Task II

```
In [11]: #In order to find the club with highest numbers of players
#use where to find the data in 2022
#group the data by club name and sort them by the count of data
highest_number_club = data_read.where(data_read['Year']=='2022').groupBy("club_name").count().sort("count",ascending=False)
print("club with highest numbers:",highest_number_club['club_name'].values)
```

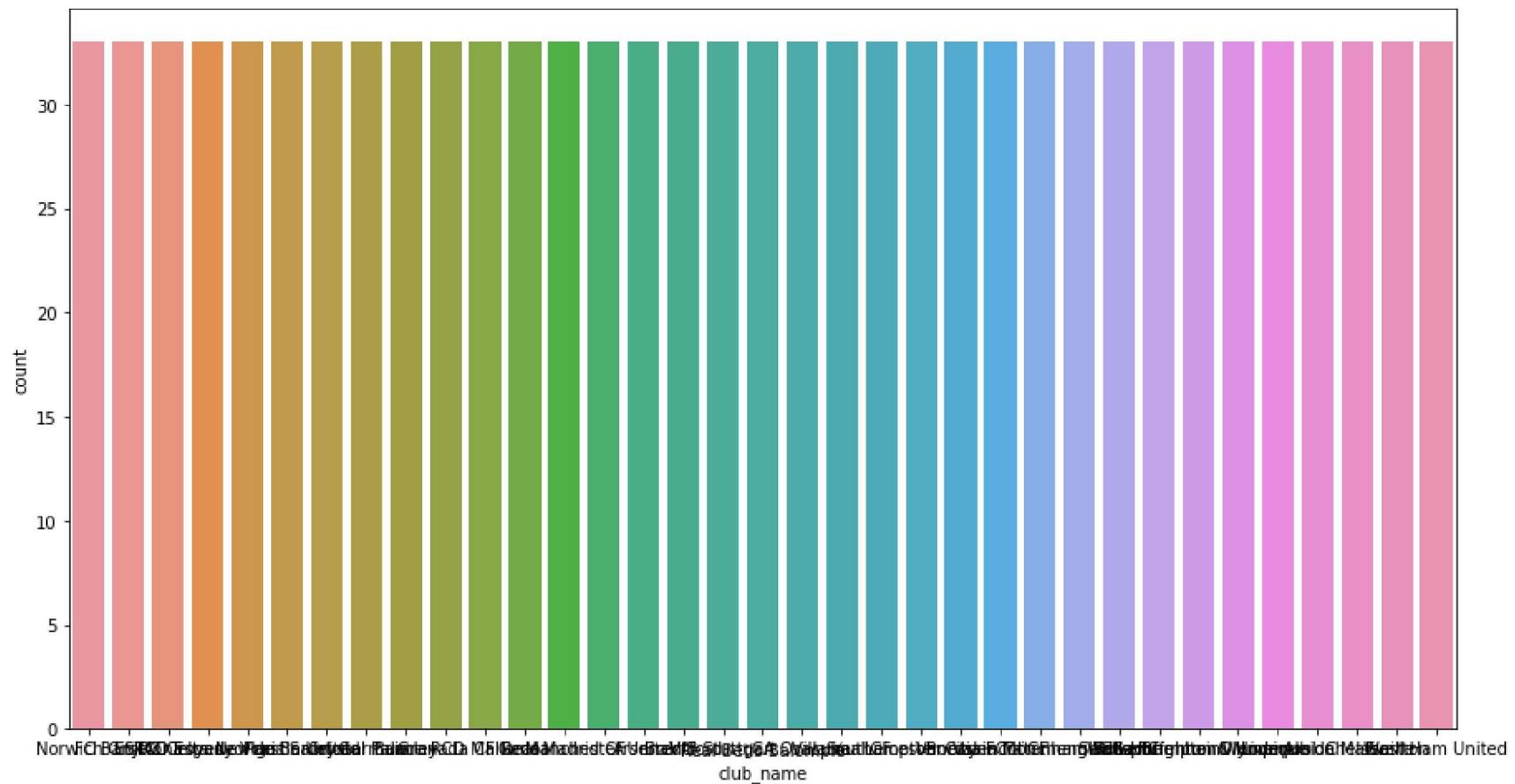
```
club with highest numbers: ['Norwich City' 'FC Barcelona' 'ESTAC Troyes' 'RC Celta de Vigo'  
'RCD Espanyol de Barcelona' 'Newcastle United' 'Paris Saint-Germain'  
'Crystal Palace' 'Burnley' 'Granada CF' 'RCD Mallorca' 'Genoa'  
'Real Madrid CF' 'Manchester United' 'Arsenal' 'Brentford'  
'VfB Stuttgart' 'Real Betis Balompié' 'CA Osasuna' 'Villarreal CF'  
'Southampton' 'Leicester City' 'Venezia FC' 'Valencia CF'  
'Borussia Mönchengladbach' 'Tottenham Hotspur' 'Sevilla FC'  
'TSG Hoffenheim' 'Wolverhampton Wanderers' 'Brighton & Hove Albion'  
'Liverpool' 'Olympique de Marseille' 'Chelsea' 'Everton'  
'West Ham United']
```

In [12]: *# this kernel shows the detailed number of players in each club and we can have the highest number from this.*

```
# we drew the barplot of the count to give a direct interpretation  
import seaborn as sns  
from IPython import display  
import matplotlib.pyplot as plt  
  
top_50_highest_number_club = data_read.where(data_read['Year']=='2022').groupBy("club_name").count().sort("count", ascending=False)  
print(top_50_highest_number_club)  
plt.figure( figsize = ( 15, 8 ) )  
sns.barplot( x="club_name", y="count", data=highest_number_club)  
plt.show()
```

	club_name	count
0	None	61
1	RC Celta de Vigo	33
2	RCD Espanyol de Barcelona	33
3	FC Barcelona	33
4	RCD Mallorca	33
5	Genoa	33
6	Real Madrid CF	33
7	ESTAC Troyes	33
8	Norwich City	33
9	VfB Stuttgart	33
10	Real Betis Balompié	33
11	Newcastle United	33
12	Arsenal	33
13	Brentford	33
14	Manchester United	33
15	Tottenham Hotspur	33
16	Sevilla FC	33
17	TSG Hoffenheim	33
18	Wolverhampton Wanderers	33
19	Brighton & Hove Albion	33
20	Liverpool	33
21	Olympique de Marseille	33
22	Chelsea	33
23	Everton	33
24	West Ham United	33
25	Levante Unión Deportiva	33
26	CA Osasuna	33
27	Paris Saint-Germain	33
28	Villarreal CF	33
29	Crystal Palace	33
30	Southampton	33
31	Leicester City	33
32	Burnley	33
33	Venezia FC	33
34	Valencia CF	33
35	Granada CF	33
36	Borussia Mönchengladbach	33
37	Lazio	32
38	SpVgg Greuther Fürth	32
39	Manchester City	32
40	Hertha BSC	32
41	RB Leipzig	32
42	1. FSV Mainz 05	32
43	VfL Wolfsburg	32

44	Cádiz CF	32
45	Roma	32
46	Atlético de Madrid	32
47	1. FC Union Berlin	31
48	FC Augsburg	31
49	Leeds United	31



```
In [14]: # the number of players older than 27 years old for each club
# we used where to find the players who are older than 27 and group the data by club_name, then we sort the data
# to find the club who has the highest number
# we sum the age for each club

import numpy as np
highest_number_over27_count = data_read.where(data_read['age']>27).groupBy("club_name").count().sort("count", ascending=False)
highest_number_over27_sum = data_read.where(data_read['age']>27).groupBy("club_name").sum('age').toPandas()
# highest_number_over27_value = highest_number_over27['age'].values
# .sort("count", ascending=False).limit(50).toPandas()
```

```

print(highest_number_over27_count)
print(highest_number_over27_sum)
average_over27 = highest_number_over27_sum['sum(age)'].values/highest_number_over27_count['count'].values
print(type(average_over27), average_over27)

```

	club_name	count
0	None	874
1	İstanbul Başakşehir FK	133
2	Jeonbuk Hyundai Motors	118
3	FC Lokomotiv Moscow	108
4	Crystal Palace	106
...	...	...
1007	Caracas Fútbol Club	3
1008	SC Freiburg II	2
1009	FC Helsingør	2
1010	FC Dordrecht	1
1011	Borussia Dortmund II	1

[1012 rows x 2 columns]

	club_name	sum(age)
0	Palermo	1105
1	CD Everton de Viña del Mar	1014
2	Shonan Bellmare	1676
3	Yeovil Town	847
4	Göztepe SK	1553
...	...	...
1007	Bury	1731
1008	Rotherham United	1945
1009	Bohemian FC	1269
1010	Middlesbrough	2064
1011	Como	219

[1012 rows x 2 columns]

```

<class 'numpy.ndarray'> [1.26430206e+00 7.62406015e+00 1.42033898e+01 ... 6.34500000e+02
2.06400000e+03 2.19000000e+02]

```

In [15]:

# we calculated the average and found the highest average

```

highest_number_over27_avg = data_read.where(data_read['age'] > 27).groupBy("club_name").avg('age').sort("avg(age)", ascending=False)
print(highest_number_over27_avg)

```

	club_name	avg(age)
0	Yokohama FC	34.703704
1	Wexford Youths	34.000000
2	Zamora Fútbol Club	33.857143
3	Centro Atlético Fénix	33.600000
4	CF Fuenlabrada	33.545455

```
In [16]: top1_highest_number_over27_avg = data_read.where(data_read['age']>27).groupBy("club_name").avg('age').sort("avg(age)", ascending=False)
print("club with highest numbers over 27:",top1_highest_number_over27_avg['club_name'].values)

club with highest numbers over 27: ['Yokohama FC']
```

```
In [17]: # most frequent nation_positions for each year
# we also used where to filter the year, and group them by nation_position, count the number of each position and sort
# them to search the highest one.

most_frequent_nation_2015 = data_read.where(data_read['Year']==2015).groupBy("nation_position").count().sort("count", ascending=False)
print("Most frequent nation position 2015:",most_frequent_nation_2015['nation_position'].values)

most_frequent_nation_2016 = data_read.where(data_read['Year']==2016).groupBy("nation_position").count().sort("count", ascending=False)
print("Most frequent nation position 2016:",most_frequent_nation_2016['nation_position'].values)

most_frequent_nation_2017 = data_read.where(data_read['Year']==2017).groupBy("nation_position").count().sort("count", ascending=False)
print("Most frequent nation position 2017:",most_frequent_nation_2017['nation_position'].values)

most_frequent_nation_2018 = data_read.where(data_read['Year']==2018).groupBy("nation_position").count().sort("count", ascending=False)
print("Most frequent nation position 2018:",most_frequent_nation_2018['nation_position'].values)

most_frequent_nation_2019 = data_read.where(data_read['Year']==2019).groupBy("nation_position").count().sort("count", ascending=False)
print("Most frequent nation position 2019:",most_frequent_nation_2019['nation_position'].values)

most_frequent_nation_2020 = data_read.where(data_read['Year']==2020).groupBy("nation_position").count().sort("count", ascending=False)
print("Most frequent nation position 2020:",most_frequent_nation_2020['nation_position'].values)

most_frequent_nation_2021 = data_read.where(data_read['Year']==2021).groupBy("nation_position").count().sort("count", ascending=False)
print("Most frequent nation position 2021:",most_frequent_nation_2021['nation_position'].values)

most_frequent_nation_2022 = data_read.where(data_read['Year']==2022).groupBy("nation_position").count().sort("count", ascending=False)
print("Most frequent nation position 2022:",most_frequent_nation_2022['nation_position'].values)
```

```
Most frequent nation position 2015: ['SUB']
Most frequent nation position 2016: ['SUB']
Most frequent nation position 2017: ['SUB']
Most frequent nation position 2018: ['SUB']
Most frequent nation position 2019: ['SUB']
Most frequent nation position 2020: ['SUB']
Most frequent nation position 2021: ['SUB']
Most frequent nation position 2022: ['SUB']
```

```
In [83]: # most_frequent_nation_2015 = data_read.where(data_read['Year']==2015).groupBy("nation_position").count().sort("count", ascending=False)
# print(most_frequent_nation_2015)
```

## Task III

### Data Engineering and cleaning

```
In [19]: # Since we only use the skillsets of the players, we dropped all the unnecessary columns
drop_list = ['player_position', 'nationality_name', 'work_rate', 'body_type', 'real_face', 'data_ID', 'short_name', 'long_name',
             'club_loaned_from', 'club_joined', 'club_contract_valid_until',
             'nationality_id', 'nation_team_id', 'nation_position', 'nation_jersey_number',
             'release_clause_eur', 'player_tags', 'player_traits', 'player_face_url',
             'club_logo_url', 'club_flag_url', 'nation_logo_url', 'nation_flag_url',
             'player_url', 'mentality_composure', 'goalkeeping_speed', 'club_position', 'league_level', 'leag
```

```
In [20]: print(len(drop_list))
```

```
33
```

```
In [21]: df_read = data_read.drop('player_positions', 'nationality_name', 'work_rate', 'body_type', 'real_face', 'data_ID', 'short_name',
                               'club_loaned_from', 'club_joined', 'club_contract_valid_until',
                               'nationality_id', 'nation_team_id', 'nation_position', 'nation_jersey_number',
                               'release_clause_eur', 'player_tags', 'player_traits', 'player_face_url',
                               'club_logo_url', 'club_flag_url', 'nation_logo_url', 'nation_flag_url',
                               'player_url', 'mentality_composure', 'goalkeeping_speed', 'club_position', 'league_level', 'leag
```

```
In [22]: print(len(data_read.columns))
print(len(df_read.columns))
```

```
112
79
```

```
In [23]: df_read.printSchema()
```

```
root
|-- sofifa_id: integer (nullable = true)
|-- overall: integer (nullable = true)
|-- potential: integer (nullable = true)
|-- value_eur: double (nullable = true)
|-- wage_eur: double (nullable = true)
|-- age: integer (nullable = true)
|-- height_cm: integer (nullable = true)
|-- weight_kg: integer (nullable = true)
|-- preferred_foot: string (nullable = true)
|-- weak_foot: integer (nullable = true)
|-- skill_moves: integer (nullable = true)
|-- international_reputation: integer (nullable = true)
|-- pace: integer (nullable = true)
|-- shooting: integer (nullable = true)
|-- passing: integer (nullable = true)
|-- dribbling: integer (nullable = true)
|-- defending: integer (nullable = true)
|-- physic: integer (nullable = true)
|-- attacking_crossing: integer (nullable = true)
|-- attacking_finishing: integer (nullable = true)
|-- attacking_heading_accuracy: integer (nullable = true)
|-- attacking_short_passing: integer (nullable = true)
|-- attacking_volleys: integer (nullable = true)
|-- skill_dribbling: integer (nullable = true)
|-- skill_curve: integer (nullable = true)
|-- skill_fk_accuracy: integer (nullable = true)
|-- skill_long_passing: integer (nullable = true)
|-- skill_ball_control: integer (nullable = true)
|-- movement_acceleration: integer (nullable = true)
|-- movement_sprint_speed: integer (nullable = true)
|-- movement_agility: integer (nullable = true)
|-- movement_reactions: integer (nullable = true)
|-- movement_balance: integer (nullable = true)
|-- power_shot_power: integer (nullable = true)
|-- power_jumping: integer (nullable = true)
|-- power_stamina: integer (nullable = true)
|-- power_strength: integer (nullable = true)
|-- power_long_shots: integer (nullable = true)
|-- mentality_aggression: integer (nullable = true)
|-- mentality_interceptions: integer (nullable = true)
|-- mentality_positioning: integer (nullable = true)
|-- mentality_vision: integer (nullable = true)
|-- mentality_penalties: integer (nullable = true)
|-- defending_marking_awareness: integer (nullable = true)
```

```
-- defending_standing_tackle: integer (nullable = true)
-- defending_sliding_tackle: integer (nullable = true)
-- goalkeeping_diving: integer (nullable = true)
-- goalkeeping_handling: integer (nullable = true)
-- goalkeeping_kicking: integer (nullable = true)
-- goalkeeping_positioning: integer (nullable = true)
-- goalkeeping_reflexes: integer (nullable = true)
-- ls: string (nullable = true)
-- st: string (nullable = true)
-- rs: string (nullable = true)
-- lw: string (nullable = true)
-- lf: string (nullable = true)
-- cf: string (nullable = true)
-- rf: string (nullable = true)
-- rw: string (nullable = true)
-- lam: string (nullable = true)
-- cam: string (nullable = true)
-- ram: string (nullable = true)
-- lm: string (nullable = true)
-- lcm: string (nullable = true)
-- cm: string (nullable = true)
-- rcm: string (nullable = true)
-- rm: string (nullable = true)
-- lwb: string (nullable = true)
-- ldm: string (nullable = true)
-- cdm: string (nullable = true)
-- rdm: string (nullable = true)
-- rwb: string (nullable = true)
-- lb: string (nullable = true)
-- lcb: string (nullable = true)
-- cb: string (nullable = true)
-- rcb: string (nullable = true)
-- rb: string (nullable = true)
-- gk: string (nullable = true)
-- Year: integer (nullable = true)
```

```
In [85]: # from pyspark.sql.functions import *

# # Notice I dropped isPenalty and isSTPlay
# null_counts_plays_df = df_read.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) \
#                                         for c in df_read.columns])

# null_counts_plays_df.show(truncate=False, vertical=True)
```

**impute 'pace','shooting','passing','dribbling','defending','physic'**

```
In [ ]: # we checked the missing values of the whole dataset and we found out that the columns with
# Larger amount of missing value: 'pace', 'shooting', 'passing', 'dribbling', 'defending', 'physic'
# Because there were many missing value in the data, we want to impute the missing value with the median value
```

```
In [24]: from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['pace']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na = df_read.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
    inputCols=columns_to_be_imputed,
    outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
    .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed = imputer.fit(df_with_filled_na).transform(df_with_filled_na)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced = df_imputed.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed = df_imputed_enhanced.withColumnRenamed("pace_imputed","pace")
```

```
In [25]: from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['shooting']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na2 = df_fully_imputed.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
    inputCols=columns_to_be_imputed,
    outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
    .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed2 = imputer.fit(df_with_filled_na2).transform(df_with_filled_na2)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced2 = df_imputed2.drop(columns_to_be_imputed[0])
```

```
# We will rename our newly imputed column with the correct name
df_fully_imputed2 = df_imputed_enhanced2.withColumnRenamed("shooting_imputed", "shooting")
```

```
In [26]: from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['passing']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na3 = df_fully_imputed2.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
    inputCols=columns_to_be_imputed,
    outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
    .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed3 = imputer.fit(df_with_filled_na3).transform(df_with_filled_na3)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced3 = df_imputed3.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed3 = df_imputed_enhanced3.withColumnRenamed("passing_imputed", "passing")
```

```
In [27]: from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['dribbling']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na4 = df_fully_imputed3.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
    inputCols=columns_to_be_imputed,
    outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
    .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed4 = imputer.fit(df_with_filled_na4).transform(df_with_filled_na4)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced4 = df_imputed4.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed4 = df_imputed_enhanced4.withColumnRenamed("dribbling_imputed", "dribbling")
```

```
In [28]: from pyspark.ml.feature import Imputer
```

```

columns_to_be_imputed = ['defending']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na5 = df_fully_imputed4.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
    inputCols=columns_to_be_imputed,
    outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
    .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed5 = imputer.fit(df_with_filled_na5).transform(df_with_filled_na5)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced5 = df_imputed5.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed5 = df_imputed_enhanced5.withColumnRenamed("defending_imputed", "defending")

```

In [29]:

```

from pyspark.ml.feature import Imputer
columns_to_be_imputed = ['physic']
value_not_in_dataset = -200

# Replace None/Missing Value with a value that can't be present in the dataset.
df_with_filled_na6 = df_fully_imputed5.fillna(-200, columns_to_be_imputed)

#Create new columns with imputed values. New columns will be suffixed with "_imputed"
imputer = Imputer (
    inputCols=columns_to_be_imputed,
    outputCols=["{}_imputed".format(c) for c in columns_to_be_imputed])\
    .setStrategy("median").setMissingValue(value_not_in_dataset)

df_imputed6 = imputer.fit(df_with_filled_na6).transform(df_with_filled_na6)
# we will drop the old column without imputation. We have only one column to be imputed
df_imputed_enhanced6 = df_imputed6.drop(columns_to_be_imputed[0])

# We will rename our newly imputed column with the correct name
df_fully_imputed6 = df_imputed_enhanced6.withColumnRenamed("physic_imputed", "physic")

```

In [30]:

```

from pyspark.sql.functions import *

# # Notice I dropped isPenalty and isSTPlay
# null_counts_plays_df = df_fully_imputed6.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) \
#         for c in df_read.columns])

```

```
# null_counts_plays_df.show(truncate=False, vertical=True)
```

## drop null

```
In [ ]: # we checked the missing value again and found out the rows with missing value  
# Then we drop the rows  
# Reason: The number of row was small compared to the entire dataset, so we just drop them
```

```
In [31]: df_read = df_fully_imputed6.dropna()
```

```
In [32]: null_counts_plays_df = df_read.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) \  
                                             for c in df_read.columns])  
  
null_counts_plays_df.show(truncate=False, vertical=True)
```

```
-RECORD 0-----  
sofifa_id | 0  
overall | 0  
potential | 0  
value_eur | 0  
wage_eur | 0  
age | 0  
height_cm | 0  
weight_kg | 0  
preferred_foot | 0  
weak_foot | 0  
skill_moves | 0  
international_reputation | 0  
attacking_crossing | 0  
attacking_finishing | 0  
attacking_heading_accuracy | 0  
attacking_short_passing | 0  
attacking_volleys | 0  
skill_dribbling | 0  
skill_curve | 0  
skill_fk_accuracy | 0  
skill_long_passing | 0  
skill_ball_control | 0  
movement_acceleration | 0  
movement_sprint_speed | 0  
movement_agility | 0  
movement_reactions | 0  
movement_balance | 0  
power_shot_power | 0  
power_jumping | 0  
power_stamina | 0  
power_strength | 0  
power_long_shots | 0  
mentality_aggression | 0  
mentality_interceptions | 0  
mentality_positioning | 0  
mentality_vision | 0  
mentality_penalties | 0  
defending_marking_awareness | 0  
defending_standing_tackle | 0  
defending_sliding_tackle | 0  
goalkeeping_diving | 0  
goalkeeping_handling | 0  
goalkeeping_kicking | 0  
goalkeeping_positioning | 0
```

goalkeeping_reflexes	0
ls	0
st	0
rs	0
lw	0
lf	0
cf	0
rf	0
rw	0
lam	0
cam	0
ram	0
lm	0
lcm	0
cm	0
rcm	0
rm	0
lwb	0
ldm	0
cdm	0
rdm	0
rwb	0
lb	0
lcb	0
cb	0
rcb	0
rb	0
gk	0
Year	0
pace	0
shooting	0
passing	0
dribbling	0
defending	0
physic	0

## Data preprocessing

### label to binary

```
In [33]: # Because there are only two values in the 'preferred_foot' column, we convert the value to 0 and 1  
label_to_binary = udf(lambda name: 0.0 if name == 'Left' else 1.0)
```

```
df_read_bi = df_read.withColumn('preferred_foot', label_to_binary(col('preferred_foot')))
```

In [34]: # df\_read\_bi.show(vertical = True)

## string to int

In [35]: *# We found the format "XX+X" in some columns and we calculated them and converted them to int*

```
def addall(string):
    if len(string) == 4:
        string = int(string[0:2]) + int(string[-1])
    elif len(string) == 2:
        string = int(string)
    elif len(string) == 6:
        string = int(string[-2:])
    return string
```

In [36]: from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType

In [37]: addallUDF = udf(lambda i: addall(i), IntegerType())

In [38]: str\_to\_int\_list = ['ls', 'st', 'rs', 'lw', 'lf', 'cf', 'rf', 'rw', 'lam', 'cam', 'ram', 'lm', 'lcm', 'cm', 'rcm', 'rm', 'lwb',
'ldm', 'cdm', 'rdm', 'rwb', 'lb', 'lcb', 'cb', 'rcb', 'rb', 'gk']

In [39]: df\_read\_int = df\_read\_bi.withColumn('ls', addallUDF(col('ls')))
df\_read\_int = df\_read\_int.withColumn('st', addallUDF(col('st')))
df\_read\_int = df\_read\_int.withColumn('rs', addallUDF(col('rs')))
df\_read\_int = df\_read\_int.withColumn('lw', addallUDF(col('lw')))
df\_read\_int = df\_read\_int.withColumn('lf', addallUDF(col('lf')))
df\_read\_int = df\_read\_int.withColumn('cf', addallUDF(col('cf')))
df\_read\_int = df\_read\_int.withColumn('rf', addallUDF(col('rf')))
df\_read\_int = df\_read\_int.withColumn('rw', addallUDF(col('rw')))
df\_read\_int = df\_read\_int.withColumn('lam', addallUDF(col('lam')))
df\_read\_int = df\_read\_int.withColumn('cam', addallUDF(col('cam')))
df\_read\_int = df\_read\_int.withColumn('ram', addallUDF(col('ram')))
df\_read\_int = df\_read\_int.withColumn('lm', addallUDF(col('lm')))
df\_read\_int = df\_read\_int.withColumn('lcm', addallUDF(col('lcm')))
df\_read\_int = df\_read\_int.withColumn('cm', addallUDF(col('cm')))
df\_read\_int = df\_read\_int.withColumn('rcm', addallUDF(col('rcm')))
df\_read\_int = df\_read\_int.withColumn('rm', addallUDF(col('rm')))
df\_read\_int = df\_read\_int.withColumn('lwb', addallUDF(col('lwb')))
df\_read\_int = df\_read\_int.withColumn('ldm', addallUDF(col('ldm')))

```
df_read_int = df_read_int.withColumn('cdm', addallUDF(col('cdm')))  
df_read_int = df_read_int.withColumn('rdm', addallUDF(col('rdm')))  
df_read_int = df_read_int.withColumn('rwb', addallUDF(col('rwb')))  
df_read_int = df_read_int.withColumn('lb', addallUDF(col('lb')))  
df_read_int = df_read_int.withColumn('lcb', addallUDF(col('lcb')))  
df_read_int = df_read_int.withColumn('cb', addallUDF(col('cb')))  
df_read_int = df_read_int.withColumn('rcb', addallUDF(col('rcb')))  
df_read_int = df_read_int.withColumn('rb', addallUDF(col('rb')))  
df_read_int = df_read_int.withColumn('gk', addallUDF(col('gk')))
```

## cast data type

```
In [40]: print(df_read_int.columns)  
  
['sofifa_id', 'overall', 'potential', 'value_eur', 'wage_eur', 'age', 'height_cm', 'weight_kg', 'preferred_foot', 'weak_foot', 'skill_moves', 'international_reputation', 'attacking_crossing', 'attacking_finishing', 'attacking_heading_accuracy', 'attacking_short_passing', 'attacking_volleys', 'skill_dribbling', 'skill_curve', 'skill_fk_accuracy', 'skill_long_passing', 'skill_ball_control', 'movement_acceleration', 'movement_sprint_speed', 'movement_agility', 'movement_reactions', 'movement_balance', 'power_shot_power', 'power_jumping', 'power_stamina', 'power_strength', 'power_long_shots', 'mentality_aggression', 'mentality_interceptions', 'mentality_positioning', 'mentality_vision', 'mentality_penalties', 'defending_marking_awareness', 'defending_standing_tackle', 'defending_sliding_tackle', 'goalkeeping_diving', 'goalkeeping_handling', 'goalkeeping_kicking', 'goalkeeping_positioning', 'goalkeeping_reflexes', 'ls', 'st', 'rs', 'lw', 'lf', 'cf', 'rf', 'rw', 'lam', 'cam', 'ram', 'lm', 'lcm', 'cm', 'rcm', 'rm', 'lwb', 'ldm', 'cdm', 'rdm', 'rwb', 'lb', 'lcb', 'cb', 'rcb', 'rb', 'gk', 'Year', 'pace', 'shooting', 'passing', 'dribbling', 'defending', 'physic']  
  
In [ ]: # we changed the type of data to DoubleType  
  
In [41]: from pyspark.sql.types import DoubleType  
for i in df_read_int.columns:  
    df_read_int = df_read_int.withColumn(i, df_read_int[i].cast(DoubleType()))  
  
In [42]: df_read_int.printSchema()
```

```
root
|-- sofi_a_id: double (nullable = true)
|-- overall: double (nullable = true)
|-- potential: double (nullable = true)
|-- value_eur: double (nullable = true)
|-- wage_eur: double (nullable = true)
|-- age: double (nullable = true)
|-- height_cm: double (nullable = true)
|-- weight_kg: double (nullable = true)
|-- preferred_foot: double (nullable = true)
|-- weak_foot: double (nullable = true)
|-- skill_moves: double (nullable = true)
|-- international_reputation: double (nullable = true)
|-- attacking_crossing: double (nullable = true)
|-- attacking_finishing: double (nullable = true)
|-- attacking_heading_accuracy: double (nullable = true)
|-- attacking_short_passing: double (nullable = true)
|-- attacking_volleys: double (nullable = true)
|-- skill_dribbling: double (nullable = true)
|-- skill_curve: double (nullable = true)
|-- skill_fk_accuracy: double (nullable = true)
|-- skill_long_passing: double (nullable = true)
|-- skill_ball_control: double (nullable = true)
|-- movement_acceleration: double (nullable = true)
|-- movement_sprint_speed: double (nullable = true)
|-- movement_agility: double (nullable = true)
|-- movement_reactions: double (nullable = true)
|-- movement_balance: double (nullable = true)
|-- power_shot_power: double (nullable = true)
|-- power_jumping: double (nullable = true)
|-- power_stamina: double (nullable = true)
|-- power_strength: double (nullable = true)
|-- power_long_shots: double (nullable = true)
|-- mentality_aggression: double (nullable = true)
|-- mentality_interceptions: double (nullable = true)
|-- mentality_positioning: double (nullable = true)
|-- mentality_vision: double (nullable = true)
|-- mentality_penalties: double (nullable = true)
|-- defending_marking_awareness: double (nullable = true)
|-- defending_standing_tackle: double (nullable = true)
|-- defending_sliding_tackle: double (nullable = true)
|-- goalkeeping_diving: double (nullable = true)
|-- goalkeeping_handling: double (nullable = true)
|-- goalkeeping_kicking: double (nullable = true)
|-- goalkeeping_positioning: double (nullable = true)
```

```
-- goalkeeping_reflexes: double (nullable = true)
-- ls: double (nullable = true)
-- st: double (nullable = true)
-- rs: double (nullable = true)
-- lw: double (nullable = true)
-- lf: double (nullable = true)
-- cf: double (nullable = true)
-- rf: double (nullable = true)
-- rw: double (nullable = true)
-- lam: double (nullable = true)
-- cam: double (nullable = true)
-- ram: double (nullable = true)
-- lm: double (nullable = true)
-- lcm: double (nullable = true)
-- cm: double (nullable = true)
-- rcm: double (nullable = true)
-- rm: double (nullable = true)
-- lwb: double (nullable = true)
-- ldm: double (nullable = true)
-- cdm: double (nullable = true)
-- rdm: double (nullable = true)
-- rwb: double (nullable = true)
-- lb: double (nullable = true)
-- lcb: double (nullable = true)
-- cb: double (nullable = true)
-- rcb: double (nullable = true)
-- rb: double (nullable = true)
-- gk: double (nullable = true)
-- Year: double (nullable = true)
-- pace: double (nullable = true)
-- shooting: double (nullable = true)
-- passing: double (nullable = true)
-- dribbling: double (nullable = true)
-- defending: double (nullable = true)
-- physic: double (nullable = true)
```

```
In [43]: df_read_features = df_read_int.withColumn('overall_new', col('overall')).drop('overall')
```

```
In [44]: df_read_int.show(5, vertical = True)
```

-RECORD 0-----

sofifa_id	190745.0
overall	69.0
potential	74.0
value_eur	1100000.0
wage_eur	10000.0
age	25.0
height_cm	191.0
weight_kg	80.0
preferred_foot	1.0
weak_foot	2.0
skill_moves	1.0
international_reputation	1.0
attacking_crossing	12.0
attacking_finishing	20.0
attacking_heading_accuracy	12.0
attacking_short_passing	22.0
attacking_volleys	20.0
skill_dribbling	18.0
skill_curve	16.0
skill_fk_accuracy	11.0
skill_long_passing	20.0
skill_ball_control	22.0
movement_acceleration	58.0
movement_sprint_speed	61.0
movement_agility	65.0
movement_reactions	62.0
movement_balance	34.0
power_shot_power	16.0
power_jumping	74.0
power_stamina	34.0
power_strength	60.0
power_long_shots	16.0
mentality_aggression	31.0
mentality_interceptions	11.0
mentality_positioning	14.0
mentality_vision	49.0
mentality_penalties	16.0
defending_marking_awareness	18.0
defending_standing_tackle	18.0
defending_sliding_tackle	16.0
goalkeeping_diving	78.0
goalkeeping_handling	59.0
goalkeeping_kicking	53.0
goalkeeping_positioning	67.0

goalkeeping_reflexes	76.0
ls	28.0
st	28.0
rs	28.0
lw	30.0
lf	29.0
cf	29.0
rf	29.0
rw	30.0
lam	32.0
cam	32.0
ram	32.0
lm	31.0
lcm	28.0
cm	28.0
rcm	28.0
rm	31.0
lwb	27.0
ldm	27.0
cdm	27.0
rdm	27.0
rwb	27.0
lb	27.0
lcb	28.0
cb	28.0
rcb	28.0
rb	27.0
gk	69.0
Year	2017.0
pace	69.0
shooting	54.0
passing	58.0
dribbling	63.0
defending	55.0
physic	66.0
-RECORD 1-----	
sofifa_id	190780.0
overall	69.0
potential	73.0
value_eur	1000000.0
wage_eur	4000.0
age	27.0
height_cm	190.0
weight_kg	98.0
preferred_foot	1.0

weak_foot	2.0
skill_moves	1.0
international_reputation	1.0
attacking_crossing	13.0
attacking_finishing	12.0
attacking_heading_accuracy	14.0
attacking_short_passing	34.0
attacking_volleys	17.0
skill_dribbling	19.0
skill_curve	13.0
skill_fk_accuracy	19.0
skill_long_passing	32.0
skill_ball_control	20.0
movement_acceleration	60.0
movement_sprint_speed	60.0
movement_agility	64.0
movement_reactions	67.0
movement_balance	52.0
power_shot_power	23.0
power_jumping	75.0
power_stamina	45.0
power_strength	86.0
power_long_shots	20.0
mentality_aggression	40.0
mentality_interceptions	28.0
mentality_positioning	13.0
mentality_vision	14.0
mentality_penalties	16.0
defending_marking_awareness	13.0
defending_standing_tackle	15.0
defending_sliding_tackle	15.0
goalkeeping_diving	72.0
goalkeeping_handling	62.0
goalkeeping_kicking	60.0
goalkeeping_positioning	64.0
goalkeeping_reflexes	75.0
ls	30.0
st	30.0
rs	30.0
lw	28.0
lf	27.0
cf	27.0
rf	27.0
rw	28.0
lam	29.0

cam	29.0
ram	29.0
lm	31.0
lcm	29.0
cm	29.0
rcm	29.0
rm	31.0
lwb	31.0
ldm	32.0
cdm	32.0
rdm	32.0
rwb	31.0
lb	31.0
lcb	33.0
cb	33.0
rcb	33.0
rb	31.0
gk	69.0
Year	2017.0
pace	69.0
shooting	54.0
passing	58.0
dribbling	63.0
defending	55.0
physic	66.0
<hr/> -RECORD 2-----	
sofifa_id	190783.0
overall	69.0
potential	69.0
value_eur	6500000.0
wage_eur	9000.0
age	31.0
height_cm	190.0
weight_kg	79.0
preferred_foot	1.0
weak_foot	2.0
skill_moves	2.0
international_reputation	1.0
attacking_crossing	57.0
attacking_finishing	33.0
attacking_heading_accuracy	64.0
attacking_short_passing	68.0
attacking_volleys	34.0
skill_dribbling	50.0
skill_curve	60.0

skill_fk_accuracy	28.0
skill_long_passing	60.0
skill_ball_control	58.0
movement_acceleration	46.0
movement_sprint_speed	34.0
movement_agility	44.0
movement_reactions	64.0
movement_balance	34.0
power_shot_power	31.0
power_jumping	63.0
power_stamina	80.0
power_strength	80.0
power_long_shots	32.0
mentality_aggression	76.0
mentality_interceptions	73.0
mentality_positioning	47.0
mentality_vision	55.0
mentality_penalties	25.0
defending_marking Awareness	65.0
defending_standing_tackle	71.0
defending_sliding_tackle	62.0
goalkeeping_diving	6.0
goalkeeping_handling	16.0
goalkeeping_kicking	7.0
goalkeeping_positioning	6.0
goalkeeping_reflexes	16.0
ls	50.0
st	50.0
rs	50.0
lw	51.0
lf	50.0
cf	50.0
rf	50.0
rw	51.0
lam	54.0
cam	54.0
ram	54.0
lm	56.0
lcm	61.0
cm	61.0
rcm	61.0
rm	56.0
lwb	64.0
ldm	68.0
cdm	68.0

rdm	68.0
rwb	64.0
lb	64.0
lcb	69.0
cb	69.0
rcb	69.0
rb	64.0
gk	18.0
Year	2017.0
pace	39.0
shooting	33.0
passing	60.0
dribbling	52.0
defending	68.0
physic	78.0
-RECORD 3-----	
sofifa_id	190792.0
overall	69.0
potential	72.0
value_eur	1000000.0
wage_eur	7000.0
age	26.0
height_cm	182.0
weight_kg	78.0
preferred_foot	1.0
weak_foot	2.0
skill_moves	1.0
international_reputation	1.0
attacking_crossing	11.0
attacking_finishing	10.0
attacking_heading_accuracy	14.0
attacking_short_passing	23.0
attacking_volleys	13.0
skill_dribbling	11.0
skill_curve	15.0
skill_fk_accuracy	14.0
skill_long_passing	20.0
skill_ball_control	22.0
movement_acceleration	53.0
movement_sprint_speed	38.0
movement_agility	56.0
movement_reactions	65.0
movement_balance	42.0
power_shot_power	22.0
power_jumping	75.0

power_stamina	18.0
power_strength	73.0
power_long_shots	18.0
mentality_aggression	17.0
mentality_interceptions	17.0
mentality_positioning	11.0
mentality_vision	32.0
mentality_penalties	14.0
defending_marking_awareness	14.0
defending_standing_tackle	18.0
defending_sliding_tackle	16.0
goalkeeping_diving	72.0
goalkeeping_handling	66.0
goalkeeping_kicking	67.0
goalkeeping_positioning	65.0
goalkeeping_reflexes	73.0
ls	26.0
st	26.0
rs	26.0
lw	25.0
lf	25.0
cf	25.0
rf	25.0
rw	25.0
lam	27.0
cam	27.0
ram	27.0
lm	25.0
lcm	25.0
cm	25.0
rcm	25.0
rm	25.0
lwb	24.0
ldm	26.0
cdm	26.0
rdm	26.0
rwb	24.0
lb	25.0
lcb	28.0
cb	28.0
rcb	28.0
rb	25.0
gk	69.0
Year	2017.0
pace	69.0

shooting	54.0
passing	58.0
dribbling	63.0
defending	55.0
physic	66.0
-RECORD 4-----	
sofifa_id	190883.0
overall	69.0
potential	69.0
value_eur	775000.0
wage_eur	6000.0
age	28.0
height_cm	175.0
weight_kg	68.0
preferred_foot	1.0
weak_foot	3.0
skill_moves	2.0
international_reputation	1.0
attacking_crossing	59.0
attacking_finishing	49.0
attacking_heading_accuracy	60.0
attacking_short_passing	67.0
attacking_volleys	48.0
skill_dribbling	57.0
skill_curve	61.0
skill_fk_accuracy	55.0
skill_long_passing	58.0
skill_ball_control	63.0
movement_acceleration	73.0
movement_sprint_speed	72.0
movement_agility	76.0
movement_reactions	71.0
movement_balance	83.0
power_shot_power	53.0
power_jumping	81.0
power_stamina	78.0
power_strength	66.0
power_long_shots	36.0
mentality_aggression	76.0
mentality_interceptions	69.0
mentality_positioning	55.0
mentality_vision	56.0
mentality_penalties	52.0
defending_marking_awareness	71.0
defending_standing_tackle	64.0

defending_sliding_tackle	68.0
goalkeeping_diving	13.0
goalkeeping_handling	9.0
goalkeeping_kicking	16.0
goalkeeping_positioning	12.0
goalkeeping_reflexes	11.0
ls	59.0
st	59.0
rs	59.0
lw	61.0
lf	59.0
cf	59.0
rf	59.0
rw	61.0
lam	61.0
cam	61.0
ram	61.0
lm	63.0
lcm	63.0
cm	63.0
rcm	63.0
rm	63.0
lwb	69.0
ldm	68.0
cdm	68.0
rdm	68.0
rwb	69.0
lb	69.0
lcb	69.0
cb	69.0
rcb	69.0
rb	69.0
gk	19.0
Year	2017.0
pace	72.0
shooting	48.0
passing	61.0
dribbling	63.0
defending	67.0
physic	72.0

only showing top 5 rows

```
In [45]: feature_cols = df_read_int.columns  
del feature_cols[2]
```

```
In [46]: from pyspark.sql import Row  
from pyspark.ml.linalg import Vectors
```

```
In [47]: len(df_read_features.columns)
```

```
Out[47]: 79
```

```
In [48]: null_counts_plays_df_2 = df_read_features.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) \  
for c in df_read_features.columns])  
  
null_counts_plays_df_2.show(truncate=False, vertical=True)
```

```
-RECORD 0-----|0
sofifa_id      |0
potential       |0
value_eur       |0
wage_eur        |0
age             |0
height_cm       |0
weight_kg       |0
preferred_foot |0
weak_foot       |0
skill_moves     |0
international_reputation |0
attacking_crossing |0
attacking_finishing |0
attacking_heading_accuracy |0
attacking_short_passing |0
attacking_volleys |0
skill_dribbling |0
skill_curve     |0
skill_fk_accuracy |0
skill_long_passing |0
skill_ball_control |0
movement_acceleration |0
movement_sprint_speed |0
movement_agility |0
movement_reactions |0
movement_balance |0
power_shot_power |0
power_jumping   |0
power_stamina   |0
power_strength   |0
power_long_shots |0
mentality_aggression |0
mentality_interceptions |0
mentality_positioning |0
mentality_vision |0
mentality_penalties |0
defending_marking_awareness |0
defending_standing_tackle |0
defending_sliding_tackle |0
goalkeeping_diving |0
goalkeeping_handling |0
goalkeeping_kicking |0
goalkeeping_positioning |0
goalkeeping_reflexes |0
```

ls	0
st	0
rs	0
lw	0
lf	0
cf	0
rf	0
rw	0
lam	0
cam	0
ram	0
lm	0
lcm	0
cm	0
rcm	0
rm	0
lwb	0
ldm	0
cdm	0
rdm	0
rwb	0
lb	0
lcb	1
cb	1
rcb	1
rb	0
gk	451
Year	0
pace	0
shooting	0
passing	0
dribbling	0
defending	0
physic	0
overall_new	0

```
In [49]: df_read_features = df_read_features.dropna()
```

```
In [50]: # we checked the data again to ensure there is no missing value in the dataset
null_counts_plays_df_2 = df_read_features.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) \
                                                 for c in df_read_features.columns])

null_counts_plays_df_2.show(truncate=False, vertical=True)
```

```
-RECORD 0-----|0
sofifa_id      |0
potential       |0
value_eur       |0
wage_eur        |0
age             |0
height_cm       |0
weight_kg       |0
preferred_foot |0
weak_foot       |0
skill_moves     |0
international_reputation |0
attacking_crossing |0
attacking_finishing |0
attacking_heading_accuracy |0
attacking_short_passing |0
attacking_volleys |0
skill_dribbling |0
skill_curve     |0
skill_fk_accuracy |0
skill_long_passing |0
skill_ball_control |0
movement_acceleration |0
movement_sprint_speed |0
movement_agility |0
movement_reactions |0
movement_balance |0
power_shot_power |0
power_jumping   |0
power_stamina   |0
power_strength   |0
power_long_shots |0
mentality_aggression |0
mentality_interceptions |0
mentality_positioning |0
mentality_vision |0
mentality_penalties |0
defending_marking_awareness |0
defending_standing_tackle |0
defending_sliding_tackle |0
goalkeeping_diving |0
goalkeeping_handling |0
goalkeeping_kicking |0
goalkeeping_positioning |0
goalkeeping_reflexes |0
```

ls	0
st	0
rs	0
lw	0
lf	0
cf	0
rf	0
rw	0
lam	0
cam	0
ram	0
lm	0
lcm	0
cm	0
rcm	0
rm	0
lwb	0
ldm	0
cdm	0
rdm	0
rwb	0
lb	0
lcb	0
cb	0
rcb	0
rb	0
gk	0
Year	0
pace	0
shooting	0
passing	0
dribbling	0
defending	0
physic	0
overall_new	0

```
In [ ]: # we assembled the features in the data to a vector so that it can be recognized by pyspark models
```

```
In [51]: def transData(data):  
    return data.rdd.map(lambda r: [r[-1], Vectors.dense(r[:-1])]).\\  
        toDF(['output', 'features'])  
  
data= transData(df_read_features)  
data.show()
```

```
+-----+-----+
|output|      features|
+-----+-----+
| 69.0|[190745.0,74.0,11...|
| 69.0|[190780.0,73.0,10...|
| 69.0|[190783.0,69.0,65...|
| 69.0|[190792.0,72.0,10...|
| 69.0|[190883.0,69.0,77...|
| 69.0|[191079.0,73.0,11...|
| 69.0|[191096.0,70.0,95...|
| 69.0|[191135.0,72.0,12...|
| 69.0|[191222.0,71.0,10...|
| 69.0|[191252.0,69.0,10...|
| 69.0|[191443.0,69.0,32...|
| 69.0|[191548.0,69.0,65...|
| 69.0|[191679.0,69.0,72...|
| 69.0|[191834.0,69.0,82...|
| 69.0|[192009.0,76.0,12...|
| 69.0|[192179.0,69.0,87...|
| 69.0|[192297.0,69.0,87...|
| 69.0|[192449.0,72.0,12...|
| 69.0|[192457.0,70.0,10...|
| 69.0|[192473.0,69.0,11...|
+-----+
only showing top 20 rows
```

```
In [ ]: # we normalized the features by using StandardScaler
```

```
In [52]: from pyspark.ml.feature import StandardScaler
Scalerizer=StandardScaler().setInputCol("features").setOutputCol("norm_features")
data_norm = Scalerizer.fit(data).transform(data)
```

```
In [53]: data_norm = data_norm.drop('features')
data_norm = data_norm.withColumn('features',col('norm_features')).drop('norm_features')
```

```
In [54]: # data_norm.select('features').show(vertical = False)
```

```
In [55]: # (trainingData, testData) = data_norm.randomSplit([0.8, 0.2])
# then we split the data into training and test dataset
(trainingData, testData) = data_norm.randomSplit([0.8, 0.2])
```

```
In [56]: trainingData.describe().show()
```

summary	output
count	111935
mean	65.68846205387055
stddev	7.070859372775934
min	40.0
max	94.0

```
In [57]: print("Training Dataset Count: " + str(trainingData.count()))
print("Test Dataset Count: " + str(testData.count()))
```

Training Dataset Count: 111935  
Test Dataset Count: 27795

## Pyspark

### Random Forest

```
In [58]: from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

rf2 = RandomForestRegressor(featuresCol = 'features', labelCol = 'output', numTrees=200, maxDepth=5)

# Train model. This also runs the indexer.
model2 = rf2.fit(trainingData)

# Make predictions.
predictions2 = model2.transform(testData)

predictions2.show(5)
```

```
+-----+-----+-----+
|output|     features|      prediction|
+-----+-----+-----+
| 41.0|[6.53690364271165...|53.49295481506021|
| 44.0|[6.41599749220878...|53.49295481506021|
| 44.0|[6.58719665904405...|53.49295481506021|
| 44.0|[6.60650222035666...|53.80888048666465|
| 45.0|[6.17380835592218...|53.55483362781141|
+-----+-----+-----+
only showing top 5 rows
```

```
In [59]: evaluator2 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse2 = evaluator2.evaluate(predictions2)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse2)
```

```
Root Mean Squared Error (RMSE) on test data = 1.58138
```

## tune the parameters

```
In [60]: from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

rf3 = RandomForestRegressor(featuresCol = 'features', labelCol = 'output', numTrees=100, maxDepth=5)

# Train model. This also runs the indexer.
model3 = rf3.fit(trainingData)

# Make predictions.
predictions3 = model3.transform(testData)

predictions3.show(5)

evaluator3 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse3 = evaluator3.evaluate(predictions3)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse3)
```

```
+-----+-----+-----+
|output|     features|      prediction|
+-----+-----+-----+
| 41.0|[6.53690364271165...|53.512072880945354|
| 44.0|[6.41599749220878...|53.512072880945354|
| 44.0|[6.58719665904405...|53.512072880945354|
| 44.0|[6.60650222035666...| 53.76495696951452|
| 45.0|[6.17380835592218...| 53.60100459946284|
+-----+-----+-----+
only showing top 5 rows
```

Root Mean Squared Error (RMSE) on test data = 1.60538

In [61]: `trainingData.show(5)`

```
+-----+-----+
|output|     features|
+-----+-----+
| 40.0|[6.30738196932836...|
| 40.0|[6.46263810504961...|
| 41.0|[6.40057623302213...|
| 42.0|[6.12293559300381...|
| 42.0|[6.22430428355967...|
+-----+-----+
only showing top 5 rows
```

In [62]: `from pyspark.ml.regression import RandomForestRegressor  
from pyspark.ml.feature import VectorIndexer  
from pyspark.ml.evaluation import RegressionEvaluator`

```
rf_1 = RandomForestRegressor(featuresCol = 'features', labelCol = 'output', numTrees=100, maxDepth=6)

# Train model. This also runs the indexer.
model_1 = rf_1.fit(trainingData)

# Make predictions.
predictions_1 = model_1.transform(testData)

predictions_1.show(5)

evaluator_1 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_1 = evaluator_1.evaluate(predictions_1)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_1)
```

```
+-----+-----+-----+
|output|    features|     prediction|
+-----+-----+-----+
| 41.0|[6.53690364271165...|52.39016251553484|
| 44.0|[6.41599749220878...|52.39016251553484|
| 44.0|[6.58719665904405...|52.39016251553484|
| 44.0|[6.60650222035666...|52.77358279353466|
| 45.0|[6.17380835592218...|52.47513371165757|
+-----+-----+-----+
only showing top 5 rows
```

Root Mean Squared Error (RMSE) on test data = 1.33804

In [ ]:

```
In [63]: from pyspark.ml.classification import LogisticRegression
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator

from sklearn.metrics import roc_curve
import pyspark.sql.functions as F
import pyspark.sql.types as T
import numpy
from matplotlib import pyplot as plt

from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

from pyspark.sql.types import Row
from pyspark.ml.linalg import Vectors
from pyspark.ml.classification import MultilayerPerceptronClassifier
```

## Linear Regression

In [ ]:

```
In [64]: from pyspark.ml.regression import LinearRegression

lrg = LinearRegression(featuresCol = 'features', labelCol = 'output')
# maxIter: int = 100 regParam=0.0
model5 = lrg.fit(trainingData)
```

```
In [65]: predictions5 = model5.transform(testData)

predictions5.show(5)

evaluator5 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse5 = evaluator5.evaluate(predictions5)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse5)
```

output	features	prediction
41.0	[6.53690364271165...]	40.753436154756855
44.0	[6.41599749220878...]	45.19423746263686
44.0	[6.58719665904405...]	48.668658566473596
44.0	[6.60650222035666...]	44.114411895386354
45.0	[6.17380835592218...]	45.62358471954002

only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 1.82633

## tune

```
In [66]: from pyspark.ml.regression import LinearRegression

# Define LinearRegression algorithm
lrg_1 = LinearRegression(featuresCol = 'features', labelCol = 'output', regParam=2.0)
# maxIter: int = 100
model_1 = lrg_1.fit(trainingData)
predictions_1 = model_1.transform(testData)

predictions_1.show(5)

evaluator_1 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_1 = evaluator_1.evaluate(predictions_1)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_1)
```

```
+-----+-----+
|output|      features|      prediction|
+-----+-----+
| 41.0|[6.53690364271165...|44.227559545991866|
| 44.0|[6.41599749220878...| 47.81024214005528|
| 44.0|[6.58719665904405...|50.707025368422734|
| 44.0|[6.60650222035666...|48.697781658265555|
| 45.0|[6.17380835592218...|48.883788636232325|
+-----+-----+
only showing top 5 rows
```

Root Mean Squared Error (RMSE) on test data = 2.14361

```
In [67]: from pyspark.ml.regression import LinearRegression

# Define LinearRegression algorithm
lrg_2 = LinearRegression(featuresCol = 'features', labelCol = 'output', maxIter=10)
model_2 = lrg_2.fit(trainingData)
predictions_2 = model_2.transform(testData)

predictions_2.show(5)

evaluator_2 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_2 = evaluator_2.evaluate(predictions_2)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_2)
```

```
+-----+-----+
|output|      features|      prediction|
+-----+-----+
| 41.0|[6.53690364271165...| 41.53811522459333|
| 44.0|[6.41599749220878...| 45.92256921728185|
| 44.0|[6.58719665904405...| 48.79175549801562|
| 44.0|[6.60650222035666...|44.533694117718795|
| 45.0|[6.17380835592218...| 45.96323644678648|
+-----+-----+
only showing top 5 rows
```

Root Mean Squared Error (RMSE) on test data = 1.86664

```
In [68]: from pyspark.ml.regression import LinearRegression

# Define LinearRegression algorithm
lrg_3 = LinearRegression(featuresCol = 'features', labelCol = 'output', maxIter=30)
model_3 = lrg_3.fit(trainingData)
```

```
predictions_3 = model_3.transform(testData)

predictions_3.show(5)

evaluator_3 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_3 = evaluator_3.evaluate(predictions_3)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_3)
```

output	features	prediction
41.0	[6.53690364271165...]	40.53397568304155
44.0	[6.41599749220878...]	45.15551922724518
44.0	[6.58719665904405...]	48.524183250218016
44.0	[6.60650222035666...]	44.02133666015817
45.0	[6.17380835592218...]	45.43579327618232

only showing top 5 rows

Root Mean Squared Error (RMSE) on test data = 1.83285

In [69]: `from pyspark.ml.regression import LinearRegression`

```
# Define LinearRegression algorithm
lrg_4 = LinearRegression(featuresCol = 'features', labelCol = 'output', maxIter=200)
model_4 = lrg_4.fit(trainingData)
predictions_4 = model_4.transform(testData)

predictions_4.show(5)

evaluator_4 = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")
rmse_4 = evaluator_4.evaluate(predictions_4)
print("Root Mean Squared Error (RMSE) on test data = %g" % rmse_4)
```

```
+-----+-----+-----+
| output | features | prediction |
+-----+-----+-----+
| 41.0|[6.53690364271165...|40.73535289532248|
| 44.0|[6.41599749220878...|45.15357021804499|
| 44.0|[6.58719665904405...|48.66151100088803|
| 44.0|[6.60650222035666...| 44.129994473203|
| 45.0|[6.17380835592218...|45.64501921754493|
+-----+-----+-----+
only showing top 5 rows
```

Root Mean Squared Error (RMSE) on test data = 1.82613

In [ ]:

## Tensorflow

In [70]: `from pyspark.sql.types import FloatType`

In [71]: `to_array = udf(lambda v: v.toArray().tolist(), ArrayType(FloatType()))`  
`df_test = testData`  
`df_validate, df_train = trainingData.randomSplit([0.5,0.5])`  
`df_train_pandas = df_train.withColumn('features', to_array('features')).toPandas()`  
`df_validate_pandas = df_validate.withColumn('features', to_array('features')).toPandas()`  
`df_test_pandas = df_test.withColumn('features', to_array('features')).toPandas()`

In [72]: `import tensorflow as tf`  
`from tensorflow import keras`  
`x_train = tf.constant(np.array(df_train_pandas['features'].values.tolist()))`  
`y_train = tf.constant(np.array(df_train_pandas['output'].values.tolist()))`  
`x_validate = tf.constant(np.array(df_validate_pandas['features'].values.tolist()))`  
`y_validate = tf.constant(np.array(df_validate_pandas['output'].values.tolist()))`

`x_test = tf.constant(np.array(df_test_pandas['features'].values.tolist()))`  
`y_test = tf.constant(np.array(df_test_pandas['output'].values.tolist()))`

```
In [73]: print(x_train)
print(y_train)

tf.Tensor(
[[6.30738211e+00 7.97132349e+00 2.59242672e-03 ... 3.16481352e+00
 3.29432011e+00 7.40527344e+00]
[6.22430420e+00 8.29017639e+00 3.45656904e-03 ... 6.43171787e+00
 3.48437715e+00 7.18747139e+00]
[6.40443134e+00 8.60902977e+00 3.45656904e-03 ... 6.43171787e+00
 3.48437715e+00 7.18747139e+00]
...
[6.02965415e-01 1.49860888e+01 1.50360746e+01 ... 9.29025936e+00
 2.09062624e+00 8.71208668e+00]
[6.02965415e-01 1.49860888e+01 1.65051174e+01 ... 9.18816853e+00
 2.09062624e+00 8.71208668e+00]
[4.58066463e+00 1.51455145e+01 1.91839581e+01 ... 9.69862270e+00
 1.52045548e+00 6.75186682e+00]], shape=(56176, 78), dtype=float64)
tf.Tensor([40. 42. 42. ... 94. 94. 94.], shape=(56176,), dtype=float64)
```

## Neural networks

```
In [74]: model_nn = keras.Sequential([
    keras.layers.Dense(78, activation='relu'),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(10, activation='relu'),
    keras.layers.Dense(1)])
```

```
In [75]: y_pred = model_nn(x_train)
model_nn.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
<hr/>		
dense (Dense)	(56176, 78)	6162
dense_1 (Dense)	(56176, 10)	790
dense_2 (Dense)	(56176, 10)	110
dense_3 (Dense)	(56176, 10)	110
dense_4 (Dense)	(56176, 1)	11
<hr/>		

Total params: 7,183

Trainable params: 7,183

Non-trainable params: 0

---

```
In [76]: print(y_pred)
print(y_train)
```

```
tf.Tensor(
[[74.732086]
[74.12477 ]
[74.16112 ]
...
[73.94921 ]
[74.17415 ]
[74.34247 ]], shape=(56176, 1), dtype=float32)
tf.Tensor([40. 42. 42. ... 94. 94. 94.], shape=(56176,), dtype=float64)
```

```
In [77]: mse = keras.losses.MeanSquaredError()
```

```
model_nn.compile(optimizer = 'adam',
      loss=mse,
      metrics=[mse])
model_nn.fit(x_train,y_train, epochs = 20,validation_data=(x_validate,y_validate),verbose = 2)

loss = mse(y_train, y_pred).numpy()
print(loss)
```

```
Epoch 1/20
1756/1756 - 6s - loss: 14.1575 - mean_squared_error: 14.1562 - val_loss: 13.5213 - val_mean_squared_error: 13.6656 - 6
s/epoch - 3ms/step
Epoch 2/20
1756/1756 - 5s - loss: 7.0432 - mean_squared_error: 7.0433 - val_loss: 6.2357 - val_mean_squared_error: 6.2705 - 5s/epo
ch - 3ms/step
Epoch 3/20
1756/1756 - 5s - loss: 5.6210 - mean_squared_error: 5.6212 - val_loss: 6.5882 - val_mean_squared_error: 6.6034 - 5s/epo
ch - 3ms/step
Epoch 4/20
1756/1756 - 5s - loss: 4.9807 - mean_squared_error: 4.9805 - val_loss: 4.2374 - val_mean_squared_error: 4.2553 - 5s/epo
ch - 3ms/step
Epoch 5/20
1756/1756 - 5s - loss: 4.7245 - mean_squared_error: 4.7238 - val_loss: 4.8499 - val_mean_squared_error: 4.8806 - 5s/epo
ch - 3ms/step
Epoch 6/20
1756/1756 - 6s - loss: 4.5663 - mean_squared_error: 4.5655 - val_loss: 4.4919 - val_mean_squared_error: 4.5089 - 6s/epo
ch - 3ms/step
Epoch 7/20
1756/1756 - 5s - loss: 4.3092 - mean_squared_error: 4.3104 - val_loss: 5.9678 - val_mean_squared_error: 5.9769 - 5s/epo
ch - 3ms/step
Epoch 8/20
1756/1756 - 5s - loss: 3.9615 - mean_squared_error: 3.9619 - val_loss: 7.5420 - val_mean_squared_error: 7.5455 - 5s/epo
ch - 3ms/step
Epoch 9/20
1756/1756 - 4s - loss: 3.6190 - mean_squared_error: 3.6186 - val_loss: 4.1830 - val_mean_squared_error: 4.1947 - 4s/epo
ch - 2ms/step
Epoch 10/20
1756/1756 - 5s - loss: 3.0706 - mean_squared_error: 3.0711 - val_loss: 2.4483 - val_mean_squared_error: 2.4569 - 5s/epo
ch - 3ms/step
Epoch 11/20
1756/1756 - 5s - loss: 2.8021 - mean_squared_error: 2.8017 - val_loss: 3.2102 - val_mean_squared_error: 3.2180 - 5s/epo
ch - 3ms/step
Epoch 12/20
1756/1756 - 5s - loss: 2.6239 - mean_squared_error: 2.6245 - val_loss: 2.4021 - val_mean_squared_error: 2.4094 - 5s/epo
ch - 3ms/step
Epoch 13/20
1756/1756 - 5s - loss: 2.5039 - mean_squared_error: 2.5045 - val_loss: 4.0864 - val_mean_squared_error: 4.0895 - 5s/epo
ch - 3ms/step
Epoch 14/20
1756/1756 - 5s - loss: 2.3757 - mean_squared_error: 2.3755 - val_loss: 2.2760 - val_mean_squared_error: 2.2826 - 5s/epo
ch - 3ms/step
Epoch 15/20
1756/1756 - 4s - loss: 2.2276 - mean_squared_error: 2.2274 - val_loss: 2.0466 - val_mean_squared_error: 2.0521 - 4s/epo
ch - 3ms/step
```

```

Epoch 16/20
1756/1756 - 5s - loss: 2.1436 - mean_squared_error: 2.1435 - val_loss: 1.8553 - val_mean_squared_error: 1.8600 - 5s/epoch - 3ms/step
Epoch 17/20
1756/1756 - 5s - loss: 1.9760 - mean_squared_error: 1.9758 - val_loss: 1.9688 - val_mean_squared_error: 1.9746 - 5s/epoch - 3ms/step
Epoch 18/20
1756/1756 - 5s - loss: 1.9285 - mean_squared_error: 1.9283 - val_loss: 1.8612 - val_mean_squared_error: 1.8662 - 5s/epoch - 3ms/step
Epoch 19/20
1756/1756 - 4s - loss: 1.8439 - mean_squared_error: 1.8442 - val_loss: 1.6813 - val_mean_squared_error: 1.6880 - 4s/epoch - 3ms/step
Epoch 20/20
1756/1756 - 5s - loss: 1.8171 - mean_squared_error: 1.8171 - val_loss: 1.6386 - val_mean_squared_error: 1.6437 - 5s/epoch - 3ms/step
132.08322

```

In [78]: `model_nn.evaluate(x_test,y_test, verbose = 2)`

```

869/869 - 1s - loss: 1.6491 - mean_squared_error: 1.6550 - 985ms/epoch - 1ms/step
[1.6490752696990967, 1.6549885272979736]

```

Out[78]:

In [ ]:

In [ ]:

In [79]: `def cross_valiation(hyper,k,s_r,x,y,logdir):`

```

    def data_split():
        for i in range(k):
            idx=tf.range(df_train_pandas.shape[0])
            splt_idx = int(s_r*df_train_pandas.shape[0])
            idx = tf.random.shuffle(idx)
            x_train, y_train = tf.gather(x, idx[:splt_idx]), tf.gather(y, idx[:splt_idx])
            x_valid, y_valid = tf.gather(x, idx[splt_idx:]), tf.gather(y, idx[splt_idx:])
        return x_train,y_train,x_valid,y_valid

    model = keras.Sequential()
    for _ in range(hparams[HP_DEPTH]):
        model.add(keras.layers.Dense(hparams[HP_WIDTH],activation='relu'))
        model.add(keras.layers.Dense(1))
        model.compile(optimizer = 'adam',
                      loss=keras.losses.MeanSquaredError(),
                      metrics=[keras.losses.MeanSquaredError(name = 'MSE')])
```

```
x_train,y_train,x_valid,y_valid = data_split()
history = model.fit(x_train, y_train, epochs=5, verbose = 2,validation_data = (x_valid, y_valid),
                     callbacks=[tf.keras.callbacks.TensorBoard(log_dir=logdir, histogram_freq=1)])
accuracy = np.mean(history.history["MSE"])
model.summary()
return accuracy
```

```
In [80]: from tensorflow.plugins.hparams import api as hp

HP_WIDTH = hp.HParam('NN_width', hp.Discrete([10,20,30]))
HP_DEPTH = hp.HParam('NN_depth', hp.Discrete([3,4,5]))


with tf.summary.create_file_writer('logs14813/hparam_tuning').as_default():
    hp.hparams_config(
        hparams=[HP_WIDTH, HP_DEPTH],
        metrics=[hp.Metric('MSE')],
    )
```

```
In [81]: import datetime
for hp_width in HP_WIDTH.domain.values:
    for hp_depth in (HP_DEPTH.domain.values):
        hparams = {
            HP_WIDTH: hp_width,
            HP_DEPTH: hp_depth,
        }
        run_name = f"run-WIDTH{int(hparams[HP_WIDTH])}-DEPTH{hparams[HP_DEPTH]}"
        print('--- Starting trial: %s' % run_name)
        print({h.name: hparams[h] for h in hparams})

        run_dir = 'logs14813/hparam_tuning/' + datetime.datetime.now().strftime("%Y%m%d-%H%M%S") + run_name
        accuracy = cross_valiation(hparams,10,0.7,x_train, y_train,run_dir)

        with tf.summary.create_file_writer(run_dir).as_default():
            hp.hparams(hparams) # record the values used in this trial
            tf.summary.scalar("MSE", accuracy, step=1)
```

```

--- Starting trial: run-WIDTH10-DEPTH3
{'NN_width': 10, 'NN_depth': 3}
Epoch 1/5
1229/1229 - 3s - loss: 69.7070 - MSE: 69.7002 - val_loss: 17.4015 - val_MSE: 17.4038 - 3s/epoch - 2ms/step
Epoch 2/5
1229/1229 - 2s - loss: 12.9095 - MSE: 12.9097 - val_loss: 11.1434 - val_MSE: 11.1429 - 2s/epoch - 2ms/step
Epoch 3/5
1229/1229 - 2s - loss: 9.6867 - MSE: 9.6868 - val_loss: 9.2053 - val_MSE: 9.2047 - 2s/epoch - 2ms/step
Epoch 4/5
1229/1229 - 2s - loss: 8.1534 - MSE: 8.1532 - val_loss: 7.4019 - val_MSE: 7.4012 - 2s/epoch - 2ms/step
Epoch 5/5
1229/1229 - 2s - loss: 7.0980 - MSE: 7.0975 - val_loss: 6.2104 - val_MSE: 6.2098 - 2s/epoch - 2ms/step
Model: "sequential_1"

```

Layer (type)	Output Shape	Param #
<hr/>		
dense_5 (Dense)	(None, 10)	790
dense_6 (Dense)	(None, 1)	11
<hr/>		
Total params: 801		
Trainable params: 801		
Non-trainable params: 0		

---

```

Epoch 1/5
1229/1229 - 3s - loss: 109.6528 - MSE: 109.6406 - val_loss: 14.2646 - val_MSE: 14.2666 - 3s/epoch - 2ms/step
Epoch 2/5
1229/1229 - 2s - loss: 10.5917 - MSE: 10.5912 - val_loss: 8.3117 - val_MSE: 8.3121 - 2s/epoch - 2ms/step
Epoch 3/5
1229/1229 - 2s - loss: 7.8114 - MSE: 7.8115 - val_loss: 7.5303 - val_MSE: 7.5294 - 2s/epoch - 2ms/step
Epoch 4/5
1229/1229 - 2s - loss: 6.2548 - MSE: 6.2548 - val_loss: 5.9674 - val_MSE: 5.9678 - 2s/epoch - 2ms/step
Epoch 5/5
1229/1229 - 2s - loss: 5.4212 - MSE: 5.4209 - val_loss: 5.0603 - val_MSE: 5.0598 - 2s/epoch - 2ms/step
Model: "sequential_1"

```

---

Layer (type)	Output Shape	Param #
<hr/>		
dense_5 (Dense)	(None, 10)	790
dense_6 (Dense)	(None, 1)	11
dense_7 (Dense)	(None, 10)	20

dense_8 (Dense)	(None, 1)	11
-----------------	-----------	----

---

Total params: 832  
 Trainable params: 832  
 Non-trainable params: 0

---

Epoch 1/5  
 1229/1229 - 3s - loss: 190.4884 - MSE: 190.4648 - val\_loss: 6.1861 - val\_MSE: 6.1857 - 3s/epoch - 3ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 5.3184 - MSE: 5.3182 - val\_loss: 5.0793 - val\_MSE: 5.0785 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 4.6593 - MSE: 4.6594 - val\_loss: 4.5452 - val\_MSE: 4.5443 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 4.2538 - MSE: 4.2538 - val\_loss: 4.2055 - val\_MSE: 4.2049 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 3s - loss: 3.9812 - MSE: 3.9811 - val\_loss: 3.8692 - val\_MSE: 3.8684 - 3s/epoch - 2ms/step  
 Model: "sequential\_1"

Layer (type)	Output Shape	Param #
<hr/>		
dense_5 (Dense)	(None, 10)	790
dense_6 (Dense)	(None, 1)	11
dense_7 (Dense)	(None, 10)	20
dense_8 (Dense)	(None, 1)	11
dense_9 (Dense)	(None, 10)	20
dense_10 (Dense)	(None, 1)	11

---

Total params: 863  
 Trainable params: 863  
 Non-trainable params: 0

---

--- Starting trial: run-WIDTH10-DEPTH4  
 {'NN\_width': 10, 'NN\_depth': 4}  
 Epoch 1/5  
 1229/1229 - 3s - loss: 522.6877 - MSE: 522.6264 - val\_loss: 27.7947 - val\_MSE: 27.8055 - 3s/epoch - 3ms/step  
 Epoch 2/5  
 1229/1229 - 3s - loss: 24.4647 - MSE: 24.4640 - val\_loss: 20.5609 - val\_MSE: 20.5685 - 3s/epoch - 2ms/step  
 Epoch 3/5

1229/1229 - 2s - loss: 17.4486 - MSE: 17.4476 - val\_loss: 14.3351 - val\_MSE: 14.3391 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 12.9500 - MSE: 12.9502 - val\_loss: 11.2775 - val\_MSE: 11.2796 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 11.0090 - MSE: 11.0100 - val\_loss: 10.6013 - val\_MSE: 10.6031 - 2s/epoch - 2ms/step  
 Model: "sequential\_2"

Layer (type)	Output Shape	Param #
<hr/>		
dense_11 (Dense)	(None, 10)	790
dense_12 (Dense)	(None, 1)	11
<hr/>		
Total params: 801		
Trainable params: 801		
Non-trainable params: 0		

---

Epoch 1/5  
 1229/1229 - 3s - loss: 458.4666 - MSE: 458.4126 - val\_loss: 63.8939 - val\_MSE: 63.9003 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 53.6201 - MSE: 53.6201 - val\_loss: 44.1140 - val\_MSE: 44.1183 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 37.9260 - MSE: 37.9247 - val\_loss: 31.1998 - val\_MSE: 31.2029 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 24.7849 - MSE: 24.7828 - val\_loss: 17.4751 - val\_MSE: 17.4755 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 13.5879 - MSE: 13.5877 - val\_loss: 10.6515 - val\_MSE: 10.6500 - 2s/epoch - 2ms/step  
 Model: "sequential\_2"

---

Layer (type)	Output Shape	Param #
<hr/>		
dense_11 (Dense)	(None, 10)	790
dense_12 (Dense)	(None, 1)	11
dense_13 (Dense)	(None, 10)	20
dense_14 (Dense)	(None, 1)	11
<hr/>		
Total params: 832		
Trainable params: 832		
Non-trainable params: 0		

---

Epoch 1/5  
 1229/1229 - 3s - loss: 695.4990 - MSE: 695.4178 - val\_loss: 57.3379 - val\_MSE: 57.3541 - 3s/epoch - 3ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 46.6103 - MSE: 46.6133 - val\_loss: 40.3770 - val\_MSE: 40.3885 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 33.7918 - MSE: 33.7906 - val\_loss: 29.0054 - val\_MSE: 29.0139 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 21.5343 - MSE: 21.5333 - val\_loss: 15.7405 - val\_MSE: 15.7458 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 11.1094 - MSE: 11.1087 - val\_loss: 10.1750 - val\_MSE: 10.1787 - 2s/epoch - 2ms/step  
 Model: "sequential\_2"

Layer (type)	Output Shape	Param #
<hr/>		
dense_11 (Dense)	(None, 10)	790
dense_12 (Dense)	(None, 1)	11
dense_13 (Dense)	(None, 10)	20
dense_14 (Dense)	(None, 1)	11
dense_15 (Dense)	(None, 10)	20
dense_16 (Dense)	(None, 1)	11
<hr/>		
Total params: 863		
Trainable params: 863		
Non-trainable params: 0		

---

Epoch 1/5  
 1229/1229 - 4s - loss: 1248.1918 - MSE: 1248.0376 - val\_loss: 45.5892 - val\_MSE: 45.5894 - 4s/epoch - 3ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 39.1724 - MSE: 39.1737 - val\_loss: 31.6969 - val\_MSE: 31.6968 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 26.6132 - MSE: 26.6122 - val\_loss: 19.4637 - val\_MSE: 19.4625 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 14.2774 - MSE: 14.2768 - val\_loss: 12.0076 - val\_MSE: 12.0094 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 10.2604 - MSE: 10.2603 - val\_loss: 10.2550 - val\_MSE: 10.2551 - 2s/epoch - 2ms/step  
 Model: "sequential\_2"

Layer (type)	Output Shape	Param #
<hr/>		

dense_11 (Dense)	(None, 10)	790
dense_12 (Dense)	(None, 1)	11
dense_13 (Dense)	(None, 10)	20
dense_14 (Dense)	(None, 1)	11
dense_15 (Dense)	(None, 10)	20
dense_16 (Dense)	(None, 1)	11
dense_17 (Dense)	(None, 10)	20
dense_18 (Dense)	(None, 1)	11

---

Total params: 894

Trainable params: 894

Non-trainable params: 0

---

--- Starting trial: run-WIDTH10-DEPTH5  
{'NN\_width': 10, 'NN\_depth': 5}  
Epoch 1/5  
1229/1229 - 3s - loss: 476.5839 - MSE: 476.5289 - val\_loss: 29.2964 - val\_MSE: 29.3007 - 3s/epoch - 2ms/step  
Epoch 2/5  
1229/1229 - 2s - loss: 25.9606 - MSE: 25.9594 - val\_loss: 21.6556 - val\_MSE: 21.6594 - 2s/epoch - 2ms/step  
Epoch 3/5  
1229/1229 - 2s - loss: 18.3905 - MSE: 18.3901 - val\_loss: 15.0761 - val\_MSE: 15.0795 - 2s/epoch - 2ms/step  
Epoch 4/5  
1229/1229 - 2s - loss: 13.1669 - MSE: 13.1660 - val\_loss: 10.7960 - val\_MSE: 10.8009 - 2s/epoch - 2ms/step  
Epoch 5/5  
1229/1229 - 2s - loss: 10.2869 - MSE: 10.2864 - val\_loss: 9.2446 - val\_MSE: 9.2490 - 2s/epoch - 2ms/step  
Model: "sequential\_3"

---

Layer (type)	Output Shape	Param #
dense_19 (Dense)	(None, 10)	790
dense_20 (Dense)	(None, 1)	11

---

Total params: 801

Trainable params: 801

Non-trainable params: 0

---

Epoch 1/5  
1229/1229 - 3s - loss: 318.6873 - MSE: 318.6536 - val\_loss: 46.5412 - val\_MSE: 46.5286 - 3s/epoch - 2ms/step  
Epoch 2/5  
1229/1229 - 2s - loss: 38.1630 - MSE: 38.1622 - val\_loss: 29.2737 - val\_MSE: 29.2646 - 2s/epoch - 2ms/step  
Epoch 3/5  
1229/1229 - 2s - loss: 21.8978 - MSE: 21.8968 - val\_loss: 13.8629 - val\_MSE: 13.8567 - 2s/epoch - 2ms/step  
Epoch 4/5  
1229/1229 - 2s - loss: 11.5420 - MSE: 11.5418 - val\_loss: 10.3700 - val\_MSE: 10.3655 - 2s/epoch - 2ms/step  
Epoch 5/5  
1229/1229 - 2s - loss: 9.5435 - MSE: 9.5431 - val\_loss: 10.1343 - val\_MSE: 10.1304 - 2s/epoch - 2ms/step  
Model: "sequential\_3"

Layer (type)	Output Shape	Param #
<hr/>		
dense_19 (Dense)	(None, 10)	790
dense_20 (Dense)	(None, 1)	11
dense_21 (Dense)	(None, 10)	20
dense_22 (Dense)	(None, 1)	11
<hr/>		
Total params: 832		
Trainable params: 832		
Non-trainable params: 0		

---

Epoch 1/5  
1229/1229 - 3s - loss: 416.6308 - MSE: 416.5794 - val\_loss: 11.6535 - val\_MSE: 11.6511 - 3s/epoch - 3ms/step  
Epoch 2/5  
1229/1229 - 2s - loss: 9.9114 - MSE: 9.9145 - val\_loss: 9.2920 - val\_MSE: 9.2899 - 2s/epoch - 2ms/step  
Epoch 3/5  
1229/1229 - 3s - loss: 8.6710 - MSE: 8.6710 - val\_loss: 8.3207 - val\_MSE: 8.3186 - 3s/epoch - 3ms/step  
Epoch 4/5  
1229/1229 - 3s - loss: 8.0196 - MSE: 8.0195 - val\_loss: 7.8673 - val\_MSE: 7.8651 - 3s/epoch - 2ms/step  
Epoch 5/5  
1229/1229 - 3s - loss: 7.5011 - MSE: 7.5012 - val\_loss: 7.2152 - val\_MSE: 7.2132 - 3s/epoch - 2ms/step  
Model: "sequential\_3"

Layer (type)	Output Shape	Param #
<hr/>		
dense_19 (Dense)	(None, 10)	790
dense_20 (Dense)	(None, 1)	11

dense_21 (Dense)	(None, 10)	20
dense_22 (Dense)	(None, 1)	11
dense_23 (Dense)	(None, 10)	20
dense_24 (Dense)	(None, 1)	11

---

Total params: 863

Trainable params: 863

Non-trainable params: 0

---

Epoch 1/5  
1229/1229 - 3s - loss: 2305.4561 - MSE: 2305.1672 - val\_loss: 48.7877 - val\_MSE: 48.8102 - 3s/epoch - 3ms/step  
Epoch 2/5  
1229/1229 - 2s - loss: 41.6964 - MSE: 41.6969 - val\_loss: 37.7130 - val\_MSE: 37.7367 - 2s/epoch - 2ms/step  
Epoch 3/5  
1229/1229 - 2s - loss: 31.7350 - MSE: 31.7346 - val\_loss: 27.1738 - val\_MSE: 27.1860 - 2s/epoch - 2ms/step  
Epoch 4/5  
1229/1229 - 3s - loss: 21.3779 - MSE: 21.3773 - val\_loss: 15.5924 - val\_MSE: 15.5978 - 3s/epoch - 2ms/step  
Epoch 5/5  
1229/1229 - 3s - loss: 13.5231 - MSE: 13.5230 - val\_loss: 11.8494 - val\_MSE: 11.8506 - 3s/epoch - 2ms/step  
Model: "sequential\_3"

Layer (type)	Output Shape	Param #
dense_19 (Dense)	(None, 10)	790
dense_20 (Dense)	(None, 1)	11
dense_21 (Dense)	(None, 10)	20
dense_22 (Dense)	(None, 1)	11
dense_23 (Dense)	(None, 10)	20
dense_24 (Dense)	(None, 1)	11
dense_25 (Dense)	(None, 10)	20
dense_26 (Dense)	(None, 1)	11

---

Total params: 894  
 Trainable params: 894  
 Non-trainable params: 0

---

Epoch 1/5  
 1229/1229 - 3s - loss: 513.7387 - MSE: 513.6749 - val\_loss: 12.0590 - val\_MSE: 12.0553 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 11.2359 - MSE: 11.2354 - val\_loss: 9.9632 - val\_MSE: 9.9608 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 9.6040 - MSE: 9.6044 - val\_loss: 9.6938 - val\_MSE: 9.6929 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 8.2819 - MSE: 8.2816 - val\_loss: 7.9046 - val\_MSE: 7.9032 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 7.0416 - MSE: 7.0414 - val\_loss: 6.7960 - val\_MSE: 6.7959 - 2s/epoch - 2ms/step  
 Model: "sequential\_3"

Layer (type)	Output Shape	Param #
<hr/>		
dense_19 (Dense)	(None, 10)	790
dense_20 (Dense)	(None, 1)	11
dense_21 (Dense)	(None, 10)	20
dense_22 (Dense)	(None, 1)	11
dense_23 (Dense)	(None, 10)	20
dense_24 (Dense)	(None, 1)	11
dense_25 (Dense)	(None, 10)	20
dense_26 (Dense)	(None, 1)	11
dense_27 (Dense)	(None, 10)	20
dense_28 (Dense)	(None, 1)	11
<hr/>		

Total params: 925  
 Trainable params: 925  
 Non-trainable params: 0

---

--- Starting trial: run-WIDTH20-DEPTH3  
 {'NN\_width': 20, 'NN\_depth': 3}

Epoch 1/5  
 1229/1229 - 3s - loss: 31.6602 - MSE: 31.6572 - val\_loss: 12.7759 - val\_MSE: 12.7738 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 10.3628 - MSE: 10.3627 - val\_loss: 8.5658 - val\_MSE: 8.5664 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 8.1465 - MSE: 8.1467 - val\_loss: 7.4850 - val\_MSE: 7.4860 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 7.0048 - MSE: 7.0048 - val\_loss: 6.7486 - val\_MSE: 6.7499 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 6.2607 - MSE: 6.2607 - val\_loss: 5.9642 - val\_MSE: 5.9654 - 2s/epoch - 2ms/step  
 Model: "sequential\_4"

Layer (type)	Output Shape	Param #
<hr/>		
dense_29 (Dense)	(None, 20)	1580
dense_30 (Dense)	(None, 1)	21
<hr/>		
Total params: 1,601		
Trainable params: 1,601		
Non-trainable params: 0		

---

Epoch 1/5  
 1229/1229 - 3s - loss: 214.7521 - MSE: 214.7276 - val\_loss: 31.8031 - val\_MSE: 31.8015 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 19.2511 - MSE: 19.2499 - val\_loss: 11.7134 - val\_MSE: 11.7114 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 10.1315 - MSE: 10.1310 - val\_loss: 10.1297 - val\_MSE: 10.1281 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 8.0900 - MSE: 8.0897 - val\_loss: 6.9480 - val\_MSE: 6.9468 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 6.5768 - MSE: 6.5765 - val\_loss: 6.3803 - val\_MSE: 6.3799 - 2s/epoch - 2ms/step  
 Model: "sequential\_4"

Layer (type)	Output Shape	Param #
<hr/>		
dense_29 (Dense)	(None, 20)	1580
dense_30 (Dense)	(None, 1)	21
dense_31 (Dense)	(None, 20)	40
dense_32 (Dense)	(None, 1)	21

```
=====
Total params: 1,662
Trainable params: 1,662
Non-trainable params: 0
```

---

```
Epoch 1/5
1229/1229 - 3s - loss: 76.4574 - MSE: 76.4488 - val_loss: 10.1545 - val_MSE: 10.1556 - 3s/epoch - 2ms/step
Epoch 2/5
1229/1229 - 2s - loss: 6.9969 - MSE: 6.9966 - val_loss: 6.2184 - val_MSE: 6.2194 - 2s/epoch - 2ms/step
Epoch 3/5
1229/1229 - 3s - loss: 5.7206 - MSE: 5.7206 - val_loss: 5.0210 - val_MSE: 5.0218 - 3s/epoch - 2ms/step
Epoch 4/5
1229/1229 - 3s - loss: 5.1095 - MSE: 5.1093 - val_loss: 4.5184 - val_MSE: 4.5189 - 3s/epoch - 2ms/step
Epoch 5/5
1229/1229 - 2s - loss: 4.6499 - MSE: 4.6496 - val_loss: 5.3976 - val_MSE: 5.3975 - 2s/epoch - 2ms/step
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
dense_29 (Dense)	(None, 20)	1580
dense_30 (Dense)	(None, 1)	21
dense_31 (Dense)	(None, 20)	40
dense_32 (Dense)	(None, 1)	21
dense_33 (Dense)	(None, 20)	40
dense_34 (Dense)	(None, 1)	21

```
=====
Total params: 1,723
Trainable params: 1,723
Non-trainable params: 0
```

---

```
--- Starting trial: run-WIDTH20-DEPTH4
{'NN_width': 20, 'NN_depth': 4}
Epoch 1/5
1229/1229 - 3s - loss: 87.3124 - MSE: 87.3030 - val_loss: 19.4831 - val_MSE: 19.4836 - 3s/epoch - 2ms/step
Epoch 2/5
1229/1229 - 2s - loss: 14.1766 - MSE: 14.1761 - val_loss: 11.4002 - val_MSE: 11.4033 - 2s/epoch - 2ms/step
Epoch 3/5
1229/1229 - 2s - loss: 9.6470 - MSE: 9.6472 - val_loss: 8.1581 - val_MSE: 8.1596 - 2s/epoch - 2ms/step
Epoch 4/5
```

1229/1229 - 2s - loss: 7.2578 - MSE: 7.2576 - val\_loss: 6.0401 - val\_MSE: 6.0404 - 2s/epoch - 2ms/step  
Epoch 5/5  
1229/1229 - 2s - loss: 5.7713 - MSE: 5.7717 - val\_loss: 4.7100 - val\_MSE: 4.7096 - 2s/epoch - 1ms/step  
Model: "sequential\_5"

Layer (type)	Output Shape	Param #
<hr/>		
dense_35 (Dense)	(None, 20)	1580
dense_36 (Dense)	(None, 1)	21
<hr/>		
Total params:	1,601	
Trainable params:	1,601	
Non-trainable params:	0	

Epoch 1/5  
1229/1229 - 3s - loss: 251.6135 - MSE: 251.5826 - val\_loss: 8.0732 - val\_MSE: 8.0725 - 3s/epoch - 2ms/step  
Epoch 2/5  
1229/1229 - 2s - loss: 6.4188 - MSE: 6.4189 - val\_loss: 5.5131 - val\_MSE: 5.5129 - 2s/epoch - 2ms/step  
Epoch 3/5  
1229/1229 - 2s - loss: 5.1755 - MSE: 5.1755 - val\_loss: 4.5808 - val\_MSE: 4.5806 - 2s/epoch - 2ms/step  
Epoch 4/5  
1229/1229 - 2s - loss: 4.4637 - MSE: 4.4635 - val\_loss: 4.1960 - val\_MSE: 4.1957 - 2s/epoch - 2ms/step  
Epoch 5/5  
1229/1229 - 2s - loss: 4.0078 - MSE: 4.0076 - val\_loss: 3.7216 - val\_MSE: 3.7214 - 2s/epoch - 2ms/step  
Model: "sequential\_5"

Layer (type)	Output Shape	Param #
<hr/>		
dense_35 (Dense)	(None, 20)	1580
dense_36 (Dense)	(None, 1)	21
dense_37 (Dense)	(None, 20)	40
dense_38 (Dense)	(None, 1)	21
<hr/>		
Total params:	1,662	
Trainable params:	1,662	
Non-trainable params:	0	

Epoch 1/5  
1229/1229 - 3s - loss: 143.1483 - MSE: 143.1321 - val\_loss: 16.5469 - val\_MSE: 16.5474 - 3s/epoch - 2ms/step

Epoch 2/5  
 1229/1229 - 2s - loss: 10.9631 - MSE: 10.9624 - val\_loss: 8.3676 - val\_MSE: 8.3688 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 7.8332 - MSE: 7.8334 - val\_loss: 7.0570 - val\_MSE: 7.0579 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 6.8916 - MSE: 6.8923 - val\_loss: 6.3456 - val\_MSE: 6.3465 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 6.1439 - MSE: 6.1437 - val\_loss: 5.7334 - val\_MSE: 5.7343 - 2s/epoch - 2ms/step  
 Model: "sequential\_5"

Layer (type)	Output Shape	Param #
<hr/>		
dense_35 (Dense)	(None, 20)	1580
dense_36 (Dense)	(None, 1)	21
dense_37 (Dense)	(None, 20)	40
dense_38 (Dense)	(None, 1)	21
dense_39 (Dense)	(None, 20)	40
dense_40 (Dense)	(None, 1)	21
<hr/>		

Total params: 1,723  
 Trainable params: 1,723  
 Non-trainable params: 0

---

Epoch 1/5  
 1229/1229 - 3s - loss: 16.6831 - MSE: 16.6816 - val\_loss: 5.7838 - val\_MSE: 5.7839 - 3s/epoch - 3ms/step  
 Epoch 2/5  
 1229/1229 - 3s - loss: 5.7062 - MSE: 5.7060 - val\_loss: 5.3774 - val\_MSE: 5.3773 - 3s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 5.3697 - MSE: 5.3693 - val\_loss: 5.0290 - val\_MSE: 5.0291 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 3s - loss: 5.0689 - MSE: 5.0688 - val\_loss: 4.6185 - val\_MSE: 4.6184 - 3s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 3s - loss: 4.7122 - MSE: 4.7122 - val\_loss: 4.4155 - val\_MSE: 4.4156 - 3s/epoch - 2ms/step  
 Model: "sequential\_5"

Layer (type)	Output Shape	Param #
<hr/>		
dense_35 (Dense)	(None, 20)	1580

dense_36 (Dense)	(None, 1)	21
dense_37 (Dense)	(None, 20)	40
dense_38 (Dense)	(None, 1)	21
dense_39 (Dense)	(None, 20)	40
dense_40 (Dense)	(None, 1)	21
dense_41 (Dense)	(None, 20)	40
dense_42 (Dense)	(None, 1)	21

---

Total params: 1,784

Trainable params: 1,784

Non-trainable params: 0

---

--- Starting trial: run-WIDTH20-DEPTH5  
{'NN\_width': 20, 'NN\_depth': 5}  
Epoch 1/5  
1229/1229 - 3s - loss: 18.4341 - MSE: 18.4346 - val\_loss: 10.3422 - val\_MSE: 10.3414 - 3s/epoch - 2ms/step  
Epoch 2/5  
1229/1229 - 2s - loss: 8.4925 - MSE: 8.4922 - val\_loss: 7.1017 - val\_MSE: 7.1017 - 2s/epoch - 1ms/step  
Epoch 3/5  
1229/1229 - 2s - loss: 6.9772 - MSE: 6.9775 - val\_loss: 7.1272 - val\_MSE: 7.1281 - 2s/epoch - 1ms/step  
Epoch 4/5  
1229/1229 - 2s - loss: 6.0618 - MSE: 6.0617 - val\_loss: 5.2572 - val\_MSE: 5.2573 - 2s/epoch - 2ms/step  
Epoch 5/5  
1229/1229 - 2s - loss: 5.4154 - MSE: 5.4151 - val\_loss: 5.0872 - val\_MSE: 5.0878 - 2s/epoch - 2ms/step  
Model: "sequential\_6"

---

Layer (type)	Output Shape	Param #
dense_43 (Dense)	(None, 20)	1580
dense_44 (Dense)	(None, 1)	21

---

Total params: 1,601

Trainable params: 1,601

Non-trainable params: 0

---

Epoch 1/5

1229/1229 - 3s - loss: 222.9749 - MSE: 222.9533 - val\_loss: 35.5399 - val\_MSE: 35.5379 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 26.8101 - MSE: 26.8090 - val\_loss: 17.7155 - val\_MSE: 17.7133 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 14.9769 - MSE: 14.9766 - val\_loss: 11.9409 - val\_MSE: 11.9392 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 10.9279 - MSE: 10.9278 - val\_loss: 8.9933 - val\_MSE: 8.9924 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 3s - loss: 8.5932 - MSE: 8.5933 - val\_loss: 7.6981 - val\_MSE: 7.6979 - 3s/epoch - 2ms/step  
 Model: "sequential\_6"

Layer (type)	Output Shape	Param #
<hr/>		
dense_43 (Dense)	(None, 20)	1580
dense_44 (Dense)	(None, 1)	21
dense_45 (Dense)	(None, 20)	40
dense_46 (Dense)	(None, 1)	21
<hr/>		
Total params: 1,662		
Trainable params: 1,662		
Non-trainable params: 0		

---

Epoch 1/5  
 1229/1229 - 4s - loss: 20.7785 - MSE: 20.7770 - val\_loss: 6.7311 - val\_MSE: 6.7292 - 4s/epoch - 3ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 6.2699 - MSE: 6.2699 - val\_loss: 5.6614 - val\_MSE: 5.6597 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 5.1215 - MSE: 5.1213 - val\_loss: 4.7349 - val\_MSE: 4.7345 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 4.2270 - MSE: 4.2271 - val\_loss: 3.6799 - val\_MSE: 3.6792 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 3.6098 - MSE: 3.6100 - val\_loss: 3.3832 - val\_MSE: 3.3830 - 2s/epoch - 2ms/step  
 Model: "sequential\_6"

---

Layer (type)	Output Shape	Param #
<hr/>		
dense_43 (Dense)	(None, 20)	1580
dense_44 (Dense)	(None, 1)	21
dense_45 (Dense)	(None, 20)	40

dense_46 (Dense)	(None, 1)	21
dense_47 (Dense)	(None, 20)	40
dense_48 (Dense)	(None, 1)	21

---

Total params: 1,723

Trainable params: 1,723

Non-trainable params: 0

---

Epoch 1/5

1229/1229 - 3s - loss: 185.4881 - MSE: 185.4695 - val\_loss: 33.1431 - val\_MSE: 33.1431 - 3s/epoch - 3ms/step

Epoch 2/5

1229/1229 - 3s - loss: 18.1768 - MSE: 18.1763 - val\_loss: 11.8693 - val\_MSE: 11.8703 - 3s/epoch - 2ms/step

Epoch 3/5

1229/1229 - 2s - loss: 10.3248 - MSE: 10.3243 - val\_loss: 9.1549 - val\_MSE: 9.1548 - 2s/epoch - 2ms/step

Epoch 4/5

1229/1229 - 2s - loss: 8.7906 - MSE: 8.7904 - val\_loss: 8.1745 - val\_MSE: 8.1740 - 2s/epoch - 2ms/step

Epoch 5/5

1229/1229 - 2s - loss: 7.7071 - MSE: 7.7068 - val\_loss: 6.8180 - val\_MSE: 6.8179 - 2s/epoch - 2ms/step

Model: "sequential\_6"

---

Layer (type)	Output Shape	Param #
dense_43 (Dense)	(None, 20)	1580
dense_44 (Dense)	(None, 1)	21
dense_45 (Dense)	(None, 20)	40
dense_46 (Dense)	(None, 1)	21
dense_47 (Dense)	(None, 20)	40
dense_48 (Dense)	(None, 1)	21
dense_49 (Dense)	(None, 20)	40
dense_50 (Dense)	(None, 1)	21

---

Total params: 1,784

Trainable params: 1,784

Non-trainable params: 0

---

Epoch 1/5  
 1229/1229 - 3s - loss: 370.6485 - MSE: 370.6024 - val\_loss: 10.5139 - val\_MSE: 10.5112 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 3s - loss: 7.9271 - MSE: 7.9268 - val\_loss: 5.7239 - val\_MSE: 5.7227 - 3s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 5.2404 - MSE: 5.2406 - val\_loss: 4.5698 - val\_MSE: 4.5689 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 3s - loss: 4.1068 - MSE: 4.1066 - val\_loss: 3.5299 - val\_MSE: 3.5292 - 3s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 3s - loss: 3.3665 - MSE: 3.3664 - val\_loss: 2.9228 - val\_MSE: 2.9223 - 3s/epoch - 2ms/step  
 Model: "sequential\_6"

---

Layer (type)	Output Shape	Param #
<hr/>		
dense_43 (Dense)	(None, 20)	1580
dense_44 (Dense)	(None, 1)	21
dense_45 (Dense)	(None, 20)	40
dense_46 (Dense)	(None, 1)	21
dense_47 (Dense)	(None, 20)	40
dense_48 (Dense)	(None, 1)	21
dense_49 (Dense)	(None, 20)	40
dense_50 (Dense)	(None, 1)	21
dense_51 (Dense)	(None, 20)	40
dense_52 (Dense)	(None, 1)	21
<hr/>		

Total params: 1,845

Trainable params: 1,845

Non-trainable params: 0

---

--- Starting trial: run-WIDTH30-DEPTH3  
 {'NN\_width': 30, 'NN\_depth': 3}  
 Epoch 1/5  
 1229/1229 - 2s - loss: 22.9941 - MSE: 22.9922 - val\_loss: 10.3455 - val\_MSE: 10.3508 - 2s/epoch - 2ms/step

Epoch 2/5  
 1229/1229 - 2s - loss: 8.4658 - MSE: 8.4654 - val\_loss: 7.4150 - val\_MSE: 7.4177 - 2s/epoch - 1ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 6.1264 - MSE: 6.1260 - val\_loss: 4.5978 - val\_MSE: 4.5987 - 2s/epoch - 1ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 4.8197 - MSE: 4.8196 - val\_loss: 4.0424 - val\_MSE: 4.0435 - 2s/epoch - 1ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 4.1383 - MSE: 4.1386 - val\_loss: 3.5956 - val\_MSE: 3.5964 - 2s/epoch - 2ms/step  
 Model: "sequential\_7"

Layer (type)	Output Shape	Param #
<hr/>		
dense_53 (Dense)	(None, 30)	2370
dense_54 (Dense)	(None, 1)	31
<hr/>		
Total params: 2,401		
Trainable params: 2,401		
Non-trainable params: 0		

Epoch 1/5  
 1229/1229 - 3s - loss: 287.0855 - MSE: 287.0576 - val\_loss: 41.3187 - val\_MSE: 41.3190 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 28.4010 - MSE: 28.3999 - val\_loss: 18.0117 - val\_MSE: 18.0154 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 14.1309 - MSE: 14.1302 - val\_loss: 11.5198 - val\_MSE: 11.5223 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 10.5202 - MSE: 10.5198 - val\_loss: 9.4835 - val\_MSE: 9.4843 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 8.0337 - MSE: 8.0334 - val\_loss: 6.7466 - val\_MSE: 6.7476 - 2s/epoch - 2ms/step  
 Model: "sequential\_7"

Layer (type)	Output Shape	Param #
<hr/>		
dense_53 (Dense)	(None, 30)	2370
dense_54 (Dense)	(None, 1)	31
dense_55 (Dense)	(None, 30)	60
dense_56 (Dense)	(None, 1)	31
<hr/>		
Total params: 2,492		

Trainable params: 2,492  
 Non-trainable params: 0

Epoch 1/5  
 1229/1229 - 3s - loss: 248.5283 - MSE: 248.5043 - val\_loss: 34.6186 - val\_MSE: 34.6245 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 21.0667 - MSE: 21.0650 - val\_loss: 10.3852 - val\_MSE: 10.3848 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 8.8700 - MSE: 8.8694 - val\_loss: 7.0444 - val\_MSE: 7.0451 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 5.8768 - MSE: 5.8766 - val\_loss: 4.3580 - val\_MSE: 4.3597 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 4.2889 - MSE: 4.2886 - val\_loss: 3.6039 - val\_MSE: 3.6045 - 2s/epoch - 2ms/step  
 Model: "sequential\_7"

Layer (type)	Output Shape	Param #
<hr/>		
dense_53 (Dense)	(None, 30)	2370
dense_54 (Dense)	(None, 1)	31
dense_55 (Dense)	(None, 30)	60
dense_56 (Dense)	(None, 1)	31
dense_57 (Dense)	(None, 30)	60
dense_58 (Dense)	(None, 1)	31
<hr/>		

Total params: 2,583  
 Trainable params: 2,583  
 Non-trainable params: 0

--- Starting trial: run-WIDTH30-DEPTH4  
 {'NN\_width': 30, 'NN\_depth': 4}  
 Epoch 1/5  
 1229/1229 - 2s - loss: 24.3315 - MSE: 24.3302 - val\_loss: 9.7458 - val\_MSE: 9.7468 - 2s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 8.8406 - MSE: 8.8401 - val\_loss: 7.9723 - val\_MSE: 7.9725 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 7.0977 - MSE: 7.0977 - val\_loss: 5.9458 - val\_MSE: 5.9461 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 6.1402 - MSE: 6.1399 - val\_loss: 5.1559 - val\_MSE: 5.1562 - 2s/epoch - 2ms/step  
 Epoch 5/5

1229/1229 - 2s - loss: 5.4999 - MSE: 5.4999 - val\_loss: 5.7218 - val\_MSE: 5.7225 - 2s/epoch - 2ms/step  
 Model: "sequential\_8"

Layer (type)	Output Shape	Param #
<hr/>		
dense_59 (Dense)	(None, 30)	2370
dense_60 (Dense)	(None, 1)	31
<hr/>		

Total params: 2,401

Trainable params: 2,401

Non-trainable params: 0

---

Epoch 1/5

1229/1229 - 3s - loss: 442.1064 - MSE: 442.0554 - val\_loss: 38.1442 - val\_MSE: 38.1435 - 3s/epoch - 2ms/step

Epoch 2/5

1229/1229 - 2s - loss: 28.7432 - MSE: 28.7421 - val\_loss: 20.1864 - val\_MSE: 20.1875 - 2s/epoch - 2ms/step

Epoch 3/5

1229/1229 - 2s - loss: 16.0373 - MSE: 16.0374 - val\_loss: 12.7755 - val\_MSE: 12.7773 - 2s/epoch - 2ms/step

Epoch 4/5

1229/1229 - 2s - loss: 11.6391 - MSE: 11.6388 - val\_loss: 11.2070 - val\_MSE: 11.2079 - 2s/epoch - 2ms/step

Epoch 5/5

1229/1229 - 2s - loss: 9.1475 - MSE: 9.1474 - val\_loss: 7.7600 - val\_MSE: 7.7619 - 2s/epoch - 2ms/step

Model: "sequential\_8"

---

Layer (type)	Output Shape	Param #
<hr/>		
dense_59 (Dense)	(None, 30)	2370
dense_60 (Dense)	(None, 1)	31
dense_61 (Dense)	(None, 30)	60
dense_62 (Dense)	(None, 1)	31
<hr/>		

Total params: 2,492

Trainable params: 2,492

Non-trainable params: 0

---

Epoch 1/5

1229/1229 - 3s - loss: 26.9687 - MSE: 26.9660 - val\_loss: 7.7166 - val\_MSE: 7.7179 - 3s/epoch - 3ms/step

Epoch 2/5

1229/1229 - 2s - loss: 7.1220 - MSE: 7.1216 - val\_loss: 6.1439 - val\_MSE: 6.1454 - 2s/epoch - 2ms/step

Epoch 3/5  
 1229/1229 - 2s - loss: 6.0351 - MSE: 6.0350 - val\_loss: 5.6020 - val\_MSE: 5.6023 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 5.1482 - MSE: 5.1481 - val\_loss: 4.4884 - val\_MSE: 4.4893 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 4.5792 - MSE: 4.5789 - val\_loss: 4.0759 - val\_MSE: 4.0766 - 2s/epoch - 2ms/step  
 Model: "sequential\_8"

Layer (type)	Output Shape	Param #
<hr/>		
dense_59 (Dense)	(None, 30)	2370
dense_60 (Dense)	(None, 1)	31
dense_61 (Dense)	(None, 30)	60
dense_62 (Dense)	(None, 1)	31
dense_63 (Dense)	(None, 30)	60
dense_64 (Dense)	(None, 1)	31
<hr/>		
Total params: 2,583		
Trainable params: 2,583		
Non-trainable params: 0		

Epoch 1/5  
 1229/1229 - 3s - loss: 15.1015 - MSE: 15.1002 - val\_loss: 4.2403 - val\_MSE: 4.2402 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 4.2167 - MSE: 4.2165 - val\_loss: 3.8036 - val\_MSE: 3.8035 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 3.9636 - MSE: 3.9636 - val\_loss: 3.6645 - val\_MSE: 3.6640 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 3.7364 - MSE: 3.7366 - val\_loss: 5.5376 - val\_MSE: 5.5368 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 3.5470 - MSE: 3.5474 - val\_loss: 3.1517 - val\_MSE: 3.1514 - 2s/epoch - 2ms/step  
 Model: "sequential\_8"

Layer (type)	Output Shape	Param #
<hr/>		
dense_59 (Dense)	(None, 30)	2370
dense_60 (Dense)	(None, 1)	31

dense_61 (Dense)	(None, 30)	60
dense_62 (Dense)	(None, 1)	31
dense_63 (Dense)	(None, 30)	60
dense_64 (Dense)	(None, 1)	31
dense_65 (Dense)	(None, 30)	60
dense_66 (Dense)	(None, 1)	31

---

=====

Total params: 2,674

Trainable params: 2,674

Non-trainable params: 0

---

--- Starting trial: run-WIDTH30-DEPTH5  
{'NN\_width': 30, 'NN\_depth': 5}  
Epoch 1/5  
1229/1229 - 2s - loss: 1510.8827 - MSE: 1510.6924 - val\_loss: 17.6088 - val\_MSE: 17.6064 - 2s/epoch - 2ms/step  
Epoch 2/5  
1229/1229 - 2s - loss: 14.2162 - MSE: 14.2162 - val\_loss: 12.1156 - val\_MSE: 12.1127 - 2s/epoch - 2ms/step  
Epoch 3/5  
1229/1229 - 2s - loss: 10.7347 - MSE: 10.7351 - val\_loss: 9.9981 - val\_MSE: 9.9953 - 2s/epoch - 1ms/step  
Epoch 4/5  
1229/1229 - 2s - loss: 9.5285 - MSE: 9.5280 - val\_loss: 9.0421 - val\_MSE: 9.0392 - 2s/epoch - 1ms/step  
Epoch 5/5  
1229/1229 - 2s - loss: 8.6157 - MSE: 8.6157 - val\_loss: 7.9695 - val\_MSE: 7.9668 - 2s/epoch - 1ms/step  
Model: "sequential\_9"

---

Layer (type)	Output Shape	Param #
dense_67 (Dense)	(None, 30)	2370
dense_68 (Dense)	(None, 1)	31

---

=====

Total params: 2,401

Trainable params: 2,401

Non-trainable params: 0

---

Epoch 1/5  
1229/1229 - 2s - loss: 221.7756 - MSE: 221.7551 - val\_loss: 42.8513 - val\_MSE: 42.8545 - 2s/epoch - 2ms/step  
Epoch 2/5

1229/1229 - 2s - loss: 31.8899 - MSE: 31.8886 - val\_loss: 22.0609 - val\_MSE: 22.0665 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 16.2120 - MSE: 16.2108 - val\_loss: 13.9488 - val\_MSE: 13.9515 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 2s - loss: 11.5623 - MSE: 11.5625 - val\_loss: 10.2698 - val\_MSE: 10.2709 - 2s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 9.4035 - MSE: 9.4035 - val\_loss: 8.7993 - val\_MSE: 8.8001 - 2s/epoch - 2ms/step  
 Model: "sequential\_9"

Layer (type)	Output Shape	Param #
<hr/>		
dense_67 (Dense)	(None, 30)	2370
dense_68 (Dense)	(None, 1)	31
dense_69 (Dense)	(None, 30)	60
dense_70 (Dense)	(None, 1)	31
<hr/>		
Total params: 2,492		
Trainable params: 2,492		
Non-trainable params: 0		

---

Epoch 1/5  
 1229/1229 - 3s - loss: 178.0667 - MSE: 178.0448 - val\_loss: 9.3654 - val\_MSE: 9.3620 - 3s/epoch - 2ms/step  
 Epoch 2/5  
 1229/1229 - 2s - loss: 8.7637 - MSE: 8.7635 - val\_loss: 8.7139 - val\_MSE: 8.7109 - 2s/epoch - 2ms/step  
 Epoch 3/5  
 1229/1229 - 2s - loss: 7.6368 - MSE: 7.6366 - val\_loss: 7.4373 - val\_MSE: 7.4349 - 2s/epoch - 2ms/step  
 Epoch 4/5  
 1229/1229 - 3s - loss: 6.7849 - MSE: 6.7848 - val\_loss: 6.4158 - val\_MSE: 6.4148 - 3s/epoch - 2ms/step  
 Epoch 5/5  
 1229/1229 - 2s - loss: 5.9909 - MSE: 5.9908 - val\_loss: 5.0049 - val\_MSE: 5.0038 - 2s/epoch - 2ms/step  
 Model: "sequential\_9"

Layer (type)	Output Shape	Param #
<hr/>		
dense_67 (Dense)	(None, 30)	2370
dense_68 (Dense)	(None, 1)	31
dense_69 (Dense)	(None, 30)	60
dense_70 (Dense)	(None, 1)	31

dense_71 (Dense)	(None, 30)	60
dense_72 (Dense)	(None, 1)	31
<hr/>		
Total params: 2,583		
Trainable params: 2,583		
Non-trainable params: 0		
<hr/>		
Epoch 1/5		
1229/1229 - 3s - loss: 220.3110 - MSE: 220.2840 - val_loss: 5.8001 - val_MSE: 5.8056 - 3s/epoch - 2ms/step		
Epoch 2/5		
1229/1229 - 2s - loss: 5.8431 - MSE: 5.8433 - val_loss: 6.1834 - val_MSE: 6.1870 - 2s/epoch - 2ms/step		
Epoch 3/5		
1229/1229 - 3s - loss: 5.5993 - MSE: 5.5992 - val_loss: 5.4977 - val_MSE: 5.5005 - 3s/epoch - 2ms/step		
Epoch 4/5		
1229/1229 - 3s - loss: 5.1799 - MSE: 5.1798 - val_loss: 4.4188 - val_MSE: 4.4218 - 3s/epoch - 2ms/step		
Epoch 5/5		
1229/1229 - 4s - loss: 4.8030 - MSE: 4.8029 - val_loss: 4.9690 - val_MSE: 4.9708 - 4s/epoch - 3ms/step		
Model: "sequential_9"		

Layer (type)	Output Shape	Param #
<hr/>		
dense_67 (Dense)	(None, 30)	2370
dense_68 (Dense)	(None, 1)	31
dense_69 (Dense)	(None, 30)	60
dense_70 (Dense)	(None, 1)	31
dense_71 (Dense)	(None, 30)	60
dense_72 (Dense)	(None, 1)	31
dense_73 (Dense)	(None, 30)	60
dense_74 (Dense)	(None, 1)	31
<hr/>		

---

Total params: 2,674  
 Trainable params: 2,674  
 Non-trainable params: 0

---

```
Epoch 1/5
1229/1229 - 4s - loss: 22.2586 - MSE: 22.2565 - val_loss: 4.8362 - val_MSE: 4.8354 - 4s/epoch - 3ms/step
Epoch 2/5
1229/1229 - 3s - loss: 4.7347 - MSE: 4.7344 - val_loss: 4.1874 - val_MSE: 4.1866 - 3s/epoch - 2ms/step
Epoch 3/5
1229/1229 - 3s - loss: 4.6565 - MSE: 4.6563 - val_loss: 5.5949 - val_MSE: 5.5945 - 3s/epoch - 2ms/step
Epoch 4/5
1229/1229 - 2s - loss: 4.5016 - MSE: 4.5014 - val_loss: 3.9336 - val_MSE: 3.9329 - 2s/epoch - 2ms/step
Epoch 5/5
1229/1229 - 3s - loss: 4.5062 - MSE: 4.5065 - val_loss: 4.5744 - val_MSE: 4.5736 - 3s/epoch - 2ms/step
Model: "sequential_9"
```

Layer (type)	Output Shape	Param #
<hr/>		
dense_67 (Dense)	(None, 30)	2370
dense_68 (Dense)	(None, 1)	31
dense_69 (Dense)	(None, 30)	60
dense_70 (Dense)	(None, 1)	31
dense_71 (Dense)	(None, 30)	60
dense_72 (Dense)	(None, 1)	31
dense_73 (Dense)	(None, 30)	60
dense_74 (Dense)	(None, 1)	31
dense_75 (Dense)	(None, 30)	60
dense_76 (Dense)	(None, 1)	31
<hr/>		

Total params: 2,765

Trainable params: 2,765

Non-trainable params: 0

---

## Linear Regression

In [82]: `tf.compat.v1.disable_eager_execution()`

```
# tf.compat.v1.enable_eager_execution()
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np

from sklearn.datasets import load_boston

def append_bias_reshape(x_train,y_train):
    n_training_samples = x_train.shape[0]
    n_dim = x_train.shape[1]
    f = np.reshape(np.c_[np.ones(n_training_samples),x_train],[n_training_samples,n_dim+1])
    l = np.reshape(y_train,[n_training_samples,1])

    return f,l

if __name__ == '__main__':

    # x,y = read_boston_data()
    # norm_features = feature_normalize(x)
    f,l = append_bias_reshape(x_train,y_train)
    n_dim = f.shape[1]

    rnd_indices = np.random.rand(len(f)) < 0.80

    x_train = f[rnd_indices]
    y_train = l[rnd_indices]
    x_test = f[~rnd_indices]
    y_test = l[~rnd_indices]

    learning_rate = 0.0001
    training_epochs = 30
    cost_history = []
    test_history = []

    X = tf.compat.v1.placeholder(tf.float32,[None,n_dim])
    Y = tf.compat.v1.placeholder(tf.float32,[None,1])
    W = tf.Variable(tf.ones([n_dim,1]))

    init = tf.compat.v1.initialize_all_variables()

    y_ = tf.matmul(X,W)
    cost = tf.reduce_mean(tf.abs(y_-Y))
```

```
training_step = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost)

sess = tf.compat.v1.Session()
sess.run(init)

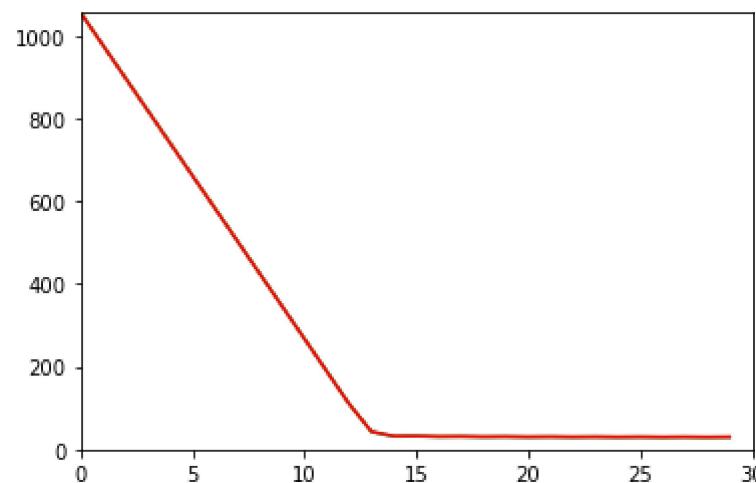
for epoch in range(training_epochs):
    sess.run(training_step, feed_dict={X:x_train,Y:y_train})
    c = sess.run(cost, feed_dict={X:x_train,Y:y_train})
    print(c)
    t = sess.run(cost, feed_dict={X:x_test,Y:y_test})
    print(t)
    cost_history.append(c)
    test_history.append(t)

plt.plot(range(len(test_history)),test_history,color = 'green')
plt.plot(range(len(cost_history)),cost_history,color = 'red')

plt.axis([0,training_epochs,0,np.max(cost_history)])
plt.show()
```

```
WARNING:tensorflow:From C:\Users\Julie\AppData\Roaming\Python\Python39\site-packages\tensorflow\python\util\tf_should_use.py:243: initialize_all_variables (from tensorflow.python.ops.variables) is deprecated and will be removed after 2017-03-02.
Instructions for updating:
Use `tf.global_variables_initializer` instead.
1056.453
1057.1117
977.57007
978.2302
898.6871
899.349
819.8042
820.46765
740.9212
741.58624
662.0383
662.70483
583.1554
583.8235
504.27246
504.9421
425.38953
426.06076
346.5066
347.17938
267.62366
268.29797
188.7407
189.41664
109.858086
110.53527
43.021408
43.00537
32.117527
31.522545
32.228374
32.035816
31.210054
30.619781
31.336395
31.12399
30.76221
30.179024
30.885654
30.656467
```

```
30.473074  
29.895824  
30.635881  
30.395695  
30.296503  
29.7233  
30.465446  
30.216637  
30.162111  
29.593527  
30.315517  
30.059908  
30.061953  
29.496569  
30.223164  
29.963722  
30.006186  
29.442467  
30.151493  
29.889187
```



```
In [86]: # tf.compat.v1.disable_eager_execution()  
# tf.compat.v1.enable_eager_execution()  
import matplotlib.pyplot as plt  
import tensorflow as tf  
import numpy as np  
  
from sklearn.datasets import load_boston  
  
def append_bias_reshape(x_train,y_train):
```

```
n_training_samples = x_train.shape[0]
n_dim = x_train.shape[1]
f = np.reshape(np.c_[np.ones(n_training_samples),x_train],[n_training_samples,n_dim+1])
l = np.reshape(y_train,[n_training_samples,1])

return f,l

if __name__ == '__main__':

#    x,y = read_boston_data()
#    norm_features = feature_normalize(x)
f,l = append_bias_reshape(x_train,y_train)
n_dim = f.shape[1]

rnd_indices = np.random.rand(len(f)) < 0.80

x_train = f[rnd_indices]
y_train = l[rnd_indices]
x_test = f[~rnd_indices]
y_test = l[~rnd_indices]


learning_rate = 0.001
training_epochs = 30
cost_history = []
test_history = []

X = tf.compat.v1.placeholder(tf.float32,[None,n_dim])
Y = tf.compat.v1.placeholder(tf.float32,[None,1])
W = tf.Variable(tf.ones([n_dim,1]))

init = tf.compat.v1.initialize_all_variables()

y_ = tf.matmul(X,W)
cost = tf.reduce_mean(tf.abs(y_-Y))
training_step = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost)

sess = tf.compat.v1.Session()
sess.run(init)

for epoch in range(training_epochs):
    sess.run(training_step,feed_dict={X:x_train,Y:y_train})
    c = sess.run(cost,feed_dict={X:x_train,Y:y_train})
```

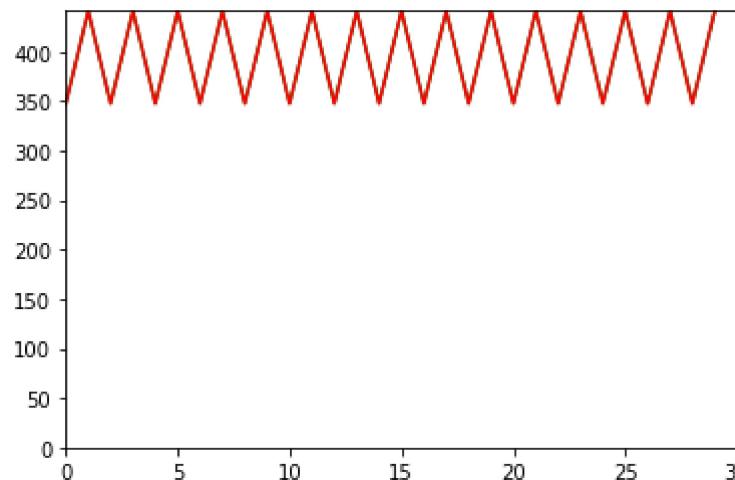
```
print(c)
t = sess.run(cost,feed_dict={X:x_test,Y:y_test})
print(t)
cost_history.append(c)
test_history.append(t)

plt.plot(range(len(test_history)),test_history,color = 'green')
plt.plot(range(len(cost_history)),cost_history,color = 'red')

plt.axis([0,training_epochs,0,np.max(cost_history)])
plt.show()
```



```
347.80347  
441.39053  
441.0418  
347.43454  
347.80347  
441.39053  
441.0418  
347.43454  
347.80347  
441.39053  
441.0418  
347.43454  
347.80347  
441.39053  
441.0418
```



```
In [87]: # tf.compat.v1.disable_eager_execution()  
# tf.compat.v1.enable_eager_execution()  
import matplotlib.pyplot as plt  
import tensorflow as tf  
import numpy as np  
  
from sklearn.datasets import load_boston  
  
def append_bias_reshape(x_train,y_train):  
    n_training_samples = x_train.shape[0]  
    n_dim = x_train.shape[1]  
    f = np.reshape(np.c_[np.ones(n_training_samples),x_train],[n_training_samples,n_dim+1])  
    l = np.reshape(y_train,[n_training_samples,1])
```

```
return f,l

if __name__ == '__main__':

#     x,y = read_boston_data()
#     norm_features = feature_normalize(x)
f,l = append_bias_reshape(x_train,y_train)
n_dim = f.shape[1]

rnd_indices = np.random.rand(len(f)) < 0.80

x_train = f[rnd_indices]
y_train = l[rnd_indices]
x_test = f[~rnd_indices]
y_test = l[~rnd_indices]

learning_rate = 0.00001
training_epochs = 300
cost_history = []
test_history = []

X = tf.compat.v1.placeholder(tf.float32,[None,n_dim])
Y = tf.compat.v1.placeholder(tf.float32,[None,1])
W = tf.Variable(tf.ones([n_dim,1]))

init = tf.compat.v1.initialize_all_variables()

y_ = tf.matmul(X,W)
cost = tf.reduce_mean(tf.abs(y_-Y))
training_step = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost)

sess = tf.compat.v1.Session()
sess.run(init)

for epoch in range(training_epochs):
    sess.run(training_step,feed_dict={X:x_train,Y:y_train})
    c = sess.run(cost,feed_dict={X:x_train,Y:y_train})
    print(c)
    t = sess.run(cost,feed_dict={X:x_test,Y:y_test})
    print(t)
    cost_history.append(c)
    test_history.append(t)
```

```
plt.plot(range(len(test_history)),test_history,color = 'green')
plt.plot(range(len(cost_history)),cost_history,color = 'red')

plt.axis([0,training_epochs,0,np.max(cost_history)])
plt.show()
```

1129.305  
1129.6372  
1121.4169  
1121.7489  
1113.5287  
1113.8607  
1105.6406  
1105.9725  
1097.7524  
1098.0844  
1089.8644  
1090.1962  
1081.9762  
1082.308  
1074.088  
1074.4198  
1066.1998  
1066.5315  
1058.3118  
1058.6433  
1050.4237  
1050.7552  
1042.5355  
1042.8671  
1034.6475  
1034.9789  
1026.7593  
1027.0906  
1018.871  
1019.20233  
1010.98303  
1011.31415  
1003.0948  
1003.42596  
995.2067  
995.5378  
987.31854  
987.6496  
979.4304  
979.7614  
971.5423  
971.8732  
963.6542  
963.985  
955.76605

956.09686  
947.87787  
948.2087  
939.9897  
940.3204  
932.1017  
932.43225  
924.21356  
924.544  
916.3253  
916.6558  
908.43726  
908.76764  
900.5491  
900.87946  
892.66095  
892.9912  
884.7729  
885.1031  
876.8847  
877.2148  
868.9965  
869.32666  
861.1084  
861.4384  
853.2203  
853.55023  
845.33215  
845.6622  
837.44403  
837.77386  
829.55597  
829.8857  
821.6677  
821.9975  
813.77966  
814.1093  
805.8914  
806.2211  
798.0034  
798.33295  
790.1151  
790.44464  
782.2271  
782.5566

774.3389  
774.6683  
766.4508  
766.7801  
758.5627  
758.8919  
750.6745  
751.0037  
742.78644  
743.11554  
734.89825  
735.22736  
727.01013  
727.3392  
719.12195  
719.451  
711.2338  
711.5628  
703.34576  
703.67456  
695.4576  
695.7863  
687.56946  
687.89825  
679.68134  
680.00995  
671.7932  
672.12177  
663.9051  
664.23364  
656.017  
656.3454  
648.12885  
648.4572  
640.24066  
640.569  
632.35254  
632.6808  
624.4644  
624.79254  
616.57623  
616.9044  
608.6881  
609.0162  
600.8

601.128  
592.91187  
593.2398  
585.0237  
585.3516  
577.1356  
577.4635  
569.24744  
569.5752  
561.3594  
561.687  
553.4712  
553.7988  
545.58307  
545.91064  
537.69495  
538.0224  
529.80676  
530.1342  
521.91864  
522.24603  
514.0306  
514.3579  
506.1424  
506.46967  
498.25427  
498.58148  
490.3661  
490.6933  
482.47797  
482.80505  
474.58984  
474.9169  
466.7017  
467.0287  
458.81357  
459.14044  
450.92538  
451.25226  
443.03726  
443.36407  
435.14905  
435.47586  
427.26096  
427.58762

419.37277  
419.6994  
411.48468  
411.8112  
403.5965  
403.92297  
395.70837  
396.03476  
387.82022  
388.14658  
379.93207  
380.25833  
372.0439  
372.37015  
364.15576  
364.48203  
356.26764  
356.59378  
348.37952  
348.70557  
340.49133  
340.81735  
332.60315  
332.9291  
324.71506  
325.04092  
316.82687  
317.1527  
308.93872  
309.26447  
301.0506  
301.37628  
293.16245  
293.48804  
285.2743  
285.59982  
277.38617  
277.71164  
269.49802  
269.82346  
261.60986  
261.93524  
253.72173  
254.04703  
245.83356

246.15883  
237.94542  
238.27063  
230.05728  
230.38242  
222.16911  
222.4942  
214.28099  
214.60597  
206.39282  
206.71779  
198.50468  
198.82954  
190.61655  
190.94138  
182.72838  
183.05316  
174.84024  
175.16496  
166.95212  
167.27675  
159.06393  
159.38853  
151.1758  
151.5003  
143.28766  
143.6121  
135.39949  
135.7239  
127.51136  
127.8357  
119.62321  
119.94748  
111.73508  
112.05928  
103.857254  
104.181046  
96.13526  
96.43865  
88.861015  
89.14738  
82.08864  
82.35444  
75.94992  
76.21351

70.42956  
70.66501  
65.30981  
65.51238  
60.438087  
60.61217  
55.74207  
55.88021  
51.21371  
51.322315  
46.944912  
47.03139  
43.05179  
43.111134  
39.633987  
39.674088  
36.762383  
36.78967  
34.48951  
34.486454  
32.765194  
32.73535  
31.49263  
31.445122  
30.57781  
30.514746  
29.945204  
29.86647  
29.511282  
29.428106  
29.214024  
29.130564  
29.0175  
28.9296  
28.889889  
28.790585  
28.80777  
28.696152  
28.754904  
28.633183  
28.721785  
28.591328  
28.701275  
28.563791  
28.687922

28.545515  
28.67933  
28.533474  
28.674025  
28.525375  
28.670565  
28.519793  
28.668154  
28.51561  
28.666506  
28.512486  
28.665386  
28.510239  
28.664627  
28.508635  
28.664038  
28.507296  
28.66358  
28.5062  
28.663212  
28.505325  
28.662884  
28.504562  
28.662594  
28.503912  
28.662344  
28.503378  
28.662102  
28.502888  
28.661877  
28.50251  
28.661655  
28.502165  
28.66144  
28.501858  
28.661219  
28.50157  
28.661005  
28.5013  
28.660791  
28.501034  
28.66058  
28.500803  
28.660368  
28.500574

28.660152  
28.500341  
28.65994  
28.500116  
28.659725  
28.49989  
28.659517  
28.499668  
28.659306  
28.499447  
28.65909  
28.499233  
28.658878  
28.499025  
28.658665  
28.498812  
28.658453  
28.4986  
28.65824  
28.49839  
28.658028  
28.498175  
28.657818  
28.497969  
28.657604  
28.497753  
28.65739  
28.497547  
28.657179  
28.497334  
28.656967  
28.497124  
28.656755  
28.496912  
28.656538  
28.496702  
28.656326  
28.496485  
28.656116  
28.496277  
28.655903  
28.496065  
28.655687  
28.495855  
28.65548

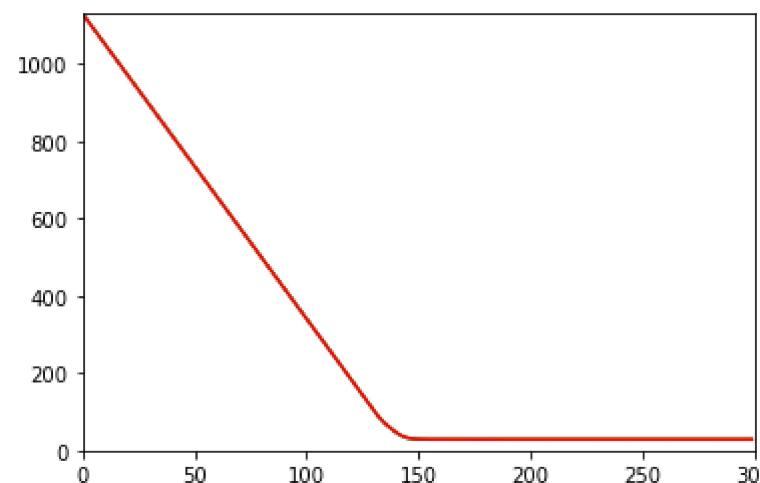
28.495651  
28.655266  
28.495426  
28.655052  
28.495218  
28.65484  
28.49501  
28.65463  
28.494802  
28.654417  
28.494583  
28.654202  
28.494371  
28.653988  
28.494165  
28.653776  
28.493958  
28.653564  
28.493736  
28.65335  
28.493525  
28.653137  
28.493319  
28.652925  
28.49311  
28.652716  
28.492899  
28.652502  
28.492682  
28.65229  
28.492476  
28.652079  
28.492262  
28.651865  
28.492052  
28.65165  
28.491837  
28.651438  
28.491623  
28.651228  
28.491417  
28.651016  
28.491205  
28.650805  
28.490992

28.650589  
28.490778  
28.650375  
28.49057  
28.650164  
28.49036  
28.649954  
28.49015  
28.649738  
28.489931  
28.649529  
28.489727  
28.649315  
28.489513  
28.6491  
28.489305  
28.648886  
28.489084  
28.648672  
28.488878  
28.648462  
28.488667  
28.64825  
28.488459  
28.648037  
28.488243  
28.647825  
28.488031  
28.647614  
28.48782  
28.647402  
28.487614  
28.647186  
28.487402  
28.646976  
28.487188  
28.646767  
28.486977  
28.646553  
28.48677  
28.646336  
28.486553  
28.646124  
28.486343  
28.645912

28.486135  
28.6457  
28.485922  
28.645489  
28.485712  
28.645275  
28.485495  
28.645063  
28.485292  
28.644852  
28.485077  
28.644638  
28.484875  
28.644426  
28.484653  
28.644213  
28.484446  
28.644  
28.484238  
28.643787  
28.484026  
28.643576  
28.483818  
28.64336  
28.483604  
28.643148  
28.483393  
28.642939  
28.483181  
28.642725  
28.48297  
28.642513  
28.482765  
28.642302  
28.482553  
28.642088  
28.482338  
28.641876  
28.482136  
28.641663  
28.481924  
28.64145  
28.481712  
28.641241  
28.481506

28.641024  
28.481293  
28.640814  
28.481077  
28.6406  
28.48087  
28.64039  
28.480663  
28.640175  
28.480446  
28.639963  
28.480242  
28.639751  
28.480032  
28.63954  
28.47982  
28.639328  
28.47961  
28.639114  
28.4794  
28.6389  
28.479195  
28.638689  
28.478975  
28.638474  
28.478765  
28.638262  
28.478558  
28.638052  
28.478348  
28.63784  
28.47814  
28.637625  
28.477922  
28.637413  
28.477713  
28.637201  
28.477499  
28.63699  
28.47729  
28.636776  
28.477074  
28.636566  
28.476866  
28.636353

```
28.476654  
28.636143  
28.476444  
28.635927  
28.476227  
28.635717  
28.47602  
28.635506  
28.475813  
28.63529  
28.475595  
28.635077  
28.475382  
28.634863  
28.475172
```



```
In [88]: # tf.compat.v1.disable_eager_execution()  
# tf.compat.v1.enable_eager_execution()  
import matplotlib.pyplot as plt  
import tensorflow as tf  
import numpy as np  
  
from sklearn.datasets import load_boston  
  
def append_bias_reshape(x_train,y_train):  
    n_training_samples = x_train.shape[0]  
    n_dim = x_train.shape[1]  
    f = np.reshape(np.c_[np.ones(n_training_samples),x_train],[n_training_samples,n_dim+1])  
    l = np.reshape(y_train,[n_training_samples,1])
```

```
return f,l

if __name__ == '__main__':

#     x,y = read_boston_data()
#     norm_features = feature_normalize(x)
f,l = append_bias_reshape(x_train,y_train)
n_dim = f.shape[1]

rnd_indices = np.random.rand(len(f)) < 0.80

x_train = f[rnd_indices]
y_train = l[rnd_indices]
x_test = f[~rnd_indices]
y_test = l[~rnd_indices]

learning_rate = 0.000001
training_epochs = 3000
cost_history = []
test_history = []

X = tf.compat.v1.placeholder(tf.float32,[None,n_dim])
Y = tf.compat.v1.placeholder(tf.float32,[None,1])
W = tf.Variable(tf.ones([n_dim,1]))

init = tf.compat.v1.initialize_all_variables()

y_ = tf.matmul(X,W)
cost = tf.reduce_mean(tf.abs(y_-Y))
training_step = tf.compat.v1.train.GradientDescentOptimizer(learning_rate).minimize(cost)

sess = tf.compat.v1.Session()
sess.run(init)

for epoch in range(training_epochs):
    sess.run(training_step,feed_dict={X:x_train,Y:y_train})
    c = sess.run(cost,feed_dict={X:x_train,Y:y_train})
    print(c)
    t = sess.run(cost,feed_dict={X:x_test,Y:y_test})
    print(t)
    cost_history.append(c)
    test_history.append(t)
```

```
plt.plot(range(len(test_history)),test_history,color = 'green')
plt.plot(range(len(cost_history)),cost_history,color = 'red')

plt.axis([0,training_epochs,0,np.max(cost_history)])
plt.show()
```

1137.4033  
1137.4084  
1136.6145  
1136.6195  
1135.8258  
1135.8307  
1135.0369  
1135.042  
1134.248  
1134.253  
1133.4594  
1133.4644  
1132.6705  
1132.6755  
1131.8817  
1131.8867  
1131.0929  
1131.0978  
1130.3041  
1130.3092  
1129.5154  
1129.5203  
1128.7266  
1128.7314  
1127.9377  
1127.9426  
1127.1488  
1127.1539  
1126.3601  
1126.3651  
1125.5712  
1125.5763  
1124.7825  
1124.7876  
1123.9935  
1123.9987  
1123.2048  
1123.2098  
1122.4161  
1122.4211  
1121.6272  
1121.6323  
1120.8385  
1120.8435  
1120.0496

1120.0546  
1119.2609  
1119.2657  
1118.4722  
1118.477  
1117.6833  
1117.6882  
1116.8944  
1116.8994  
1116.1056  
1116.1106  
1115.3168  
1115.3218  
1114.5281  
1114.5331  
1113.7393  
1113.7441  
1112.9504  
1112.9554  
1112.1616  
1112.1665  
1111.3728  
1111.3778  
1110.5841  
1110.589  
1109.7953  
1109.8  
1109.0065  
1109.0115  
1108.2177  
1108.2225  
1107.4288  
1107.4337  
1106.64  
1106.645  
1105.8513  
1105.8561  
1105.0625  
1105.0674  
1104.2736  
1104.2786  
1103.4849  
1103.4897  
1102.696  
1102.7009

1101.9073  
1101.9121  
1101.1183  
1101.1233  
1100.3297  
1100.3345  
1099.5408  
1099.5457  
1098.752  
1098.7567  
1097.9631  
1097.9681  
1097.1744  
1097.1792  
1096.3855  
1096.3905  
1095.5968  
1095.6017  
1094.8081  
1094.8129  
1094.0192  
1094.024  
1093.2303  
1093.2354  
1092.4415  
1092.4465  
1091.6528  
1091.6576  
1090.864  
1090.8688  
1090.0752  
1090.0801  
1089.2864  
1089.2913  
1088.4976  
1088.5024  
1087.7087  
1087.7137  
1086.92  
1086.9249  
1086.1312  
1086.1361  
1085.3424  
1085.3472  
1084.5536

1084.5583  
1083.7648  
1083.7695  
1082.976  
1082.9808  
1082.1873  
1082.1919  
1081.3983  
1081.4031  
1080.6095  
1080.6144  
1079.8208  
1079.8257  
1079.032  
1079.0367  
1078.2432  
1078.2479  
1077.4543  
1077.4592  
1076.6656  
1076.6704  
1075.8767  
1075.8815  
1075.0879  
1075.0927  
1074.2992  
1074.3038  
1073.5103  
1073.5153  
1072.7216  
1072.7263  
1071.9327  
1071.9375  
1071.1439  
1071.1487  
1070.355  
1070.36  
1069.5663  
1069.571  
1068.7775  
1068.7823  
1067.9888  
1067.9935  
1067.2  
1067.2047

1066.4111  
1066.416  
1065.6223  
1065.627  
1064.8335  
1064.8384  
1064.0448  
1064.0496  
1063.2559  
1063.2606  
1062.467  
1062.4718  
1061.6783  
1061.6832  
1060.8895  
1060.8943  
1060.1007  
1060.1055  
1059.3118  
1059.3167  
1058.5232  
1058.5278  
1057.7343  
1057.739  
1056.9454  
1056.9503  
1056.1567  
1056.1614  
1055.3679  
1055.3727  
1054.579  
1054.5839  
1053.7903  
1053.7949  
1053.0015  
1053.0062  
1052.2126  
1052.2174  
1051.4238  
1051.4286  
1050.635  
1050.6398  
1049.8462  
1049.8508  
1049.0575

1049.0621  
1048.2687  
1048.2733  
1047.4799  
1047.4845  
1046.691  
1046.6958  
1045.9023  
1045.907  
1045.1135  
1045.1182  
1044.3247  
1044.3293  
1043.5359  
1043.5404  
1042.747  
1042.7518  
1041.9583  
1041.963  
1041.1694  
1041.1742  
1040.3807  
1040.3854  
1039.5919  
1039.5967  
1038.8031  
1038.8077  
1038.0143  
1038.0189  
1037.2255  
1037.23  
1036.4368  
1036.4413  
1035.6478  
1035.6525  
1034.859  
1034.8636  
1034.0702  
1034.075  
1033.2815  
1033.286  
1032.4927  
1032.4973  
1031.704  
1031.7085

1030.9149  
1030.9198  
1030.1262  
1030.1309  
1029.3374  
1029.342  
1028.5486  
1028.5532  
1027.7599  
1027.7644  
1026.971  
1026.9756  
1026.1823  
1026.1868  
1025.3934  
1025.398  
1024.6046  
1024.6093  
1023.8158  
1023.82043  
1023.02704  
1023.0317  
1022.2382  
1022.24274  
1021.4494  
1021.454  
1020.66064  
1020.6652  
1019.87177  
1019.87634  
1019.083  
1019.08765  
1018.2941  
1018.2988  
1017.50543  
1017.5101  
1016.7165  
1016.72125  
1015.92786  
1015.9324  
1015.13885  
1015.14355  
1014.3503  
1014.35474  
1013.5613

1013.5659  
1012.7725  
1012.77704  
1011.98376  
1011.9884  
1011.1949  
1011.1996  
1010.4062  
1010.4107  
1009.6173  
1009.62195  
1008.8286  
1008.83307  
1008.03973  
1008.0444  
1007.2509  
1007.25543  
1006.4622  
1006.46674  
1005.67334  
1005.67786  
1004.8845  
1004.8891  
1004.09576  
1004.1003  
1003.30695  
1003.3115  
1002.5181  
1002.5227  
1001.7294  
1001.73395  
1000.94055  
1000.9451  
1000.15173  
1000.1563  
999.363  
999.36743  
998.5741  
998.5788  
997.7854  
997.7897  
996.9966  
997.00104  
996.2077  
996.2123

995.4189  
995.4235  
994.6302  
994.6347  
993.8413  
993.84576  
993.0524  
993.057  
992.26373  
992.2682  
991.475  
991.4794  
990.6861  
990.6907  
989.89734  
989.90186  
989.1085  
989.11304  
988.31976  
988.32416  
987.5308  
987.5354  
986.74207  
986.7466  
985.95325  
985.9578  
985.1645  
985.1689  
984.3757  
984.3802  
983.58685  
983.5913  
982.7981  
982.80255  
982.0093  
982.0137  
981.22046  
981.225  
980.4317  
980.43616  
979.6429  
979.64734  
978.85406  
978.8586  
978.0653

978.06964  
977.2764  
977.2809  
976.48773  
976.4921  
975.69885  
975.7033  
974.9101  
974.91455  
974.12134  
974.1256  
973.3325  
973.3369  
972.54364  
972.548  
971.75494  
971.7594  
970.96606  
970.97046  
970.17737  
970.1818  
969.3884  
969.3929  
968.59955  
968.6041  
967.8108  
967.8153  
967.022  
967.02637  
966.2332  
966.2376  
965.44446  
965.4488  
964.6556  
964.66003  
963.8668  
963.8713  
963.078  
963.08246  
962.2892  
962.29364  
961.5005  
961.50476  
960.7116  
960.7161

959.9228  
959.9271  
959.1339  
959.1384  
958.3452  
958.3496  
957.55634  
957.56085  
956.76764  
956.7721  
955.9788  
955.9832  
955.19006  
955.19446  
954.40125  
954.4055  
953.6124  
953.61676  
952.82367  
952.82794  
952.03485  
952.03906  
951.2461  
951.2504  
950.4573  
950.4615  
949.66846  
949.6727  
948.8795  
948.88403  
948.0908  
948.0951  
947.302  
947.3064  
946.5131  
946.5175  
945.7244  
945.72876  
944.9356  
944.93994  
944.1467  
944.15106  
943.3579  
943.3623  
942.56915

942.5735  
941.78046  
941.7847  
940.9915  
940.9959  
940.20276  
940.2071  
939.41394  
939.4182  
938.6252  
938.62946  
937.83636  
937.84076  
937.04755  
937.0519  
936.2588  
936.26306  
935.47  
935.47424  
934.68115  
934.68555  
933.8923  
933.89667  
933.1036  
933.1078  
932.3148  
932.31903  
931.526  
931.5302  
930.7372  
930.7414  
929.9483  
929.95264  
929.15955  
929.1639  
928.3707  
928.37506  
927.58185  
927.5862  
926.79315  
926.79736  
926.0043  
926.0086  
925.2155  
925.2198

924.42676  
924.431  
923.63794  
923.6422  
922.8492  
922.85333  
922.06024  
922.0645  
921.2715  
921.27576  
920.48267  
920.48694  
919.69403  
919.6982  
918.9051  
918.90936  
918.1163  
918.12054  
917.3275  
917.33167  
916.53876  
916.543  
915.7499  
915.7542  
914.9611  
914.96533  
914.1723  
914.1765  
913.38354  
913.3877  
912.5947  
912.5988  
911.8059  
911.8102  
911.01715  
911.02136  
910.22833  
910.2325  
909.4395  
909.44366  
908.65076  
908.65485  
907.8619  
907.86615  
907.07306

907.07733  
906.28436  
906.2883  
905.49554  
905.49963  
904.70667  
904.7108  
903.91785  
903.922  
903.12897  
903.1333  
902.3402  
902.3445  
901.5514  
901.55566  
900.76263  
900.7669  
899.9739  
899.9781  
899.185  
899.18915  
898.39624  
898.4004  
897.6075  
897.61163  
896.8186  
896.8228  
896.02985  
896.034  
895.241  
895.2451  
894.4522  
894.4565  
893.66345  
893.66766  
892.87463  
892.8787  
892.0859  
892.08997  
891.297  
891.30115  
890.50824  
890.51245  
889.7195  
889.7235

888.93066  
888.9348  
888.14185  
888.14594  
887.3531  
887.3571  
886.5642  
886.56836  
885.7755  
885.77954  
884.9866  
884.9908  
884.1979  
884.20197  
883.409  
883.4131  
882.6203  
882.62445  
881.8314  
881.8355  
881.0427  
881.04675  
880.25385  
880.25793  
879.465  
879.4691  
878.6762  
878.68024  
877.88745  
877.89154  
877.0986  
877.10266  
876.3099  
876.3139  
875.52106  
875.5251  
874.7322  
874.7363  
873.9435  
873.9475  
873.1546  
873.1587  
872.3657  
872.3698  
871.57697

871.5811  
870.7882  
870.79224  
869.9994  
870.0035  
869.2106  
869.2146  
868.4218  
868.4259  
867.633  
867.6371  
866.84424  
866.84827  
866.0553  
866.0595  
865.2666  
865.27057  
864.47784  
864.48193  
863.689  
863.693  
862.9002  
862.90424  
862.1113  
862.1154  
861.32245  
861.32654  
860.53375  
860.53784  
859.74493  
859.7489  
858.95605  
858.96014  
858.16736  
858.1714  
857.3785  
857.38257  
856.5898  
856.5937  
855.8009  
855.805  
855.0122  
855.01624  
854.2233  
854.2273

853.4345  
853.43854  
852.6457  
852.6497  
851.85693  
851.86096  
851.0681  
851.072  
850.2793  
850.2834  
849.49054  
849.49445  
848.7017  
848.7057  
847.91296  
847.9169  
847.12415  
847.1281  
846.3353  
846.3392  
845.5466  
845.55054  
844.75775  
844.7616  
843.96893  
843.97284  
843.18005  
843.1841  
842.3913  
842.39526  
841.60254  
841.6065  
840.81366  
840.8177  
840.02496  
840.0289  
839.2361  
839.24  
838.4473  
838.4512  
837.6585  
837.6624  
836.8697  
836.87366  
836.08093

836.08484  
835.2921  
835.296  
834.5033  
834.50714  
833.7144  
833.71844  
832.9257  
832.9297  
832.13684  
832.14075  
831.348  
831.352  
830.55927  
830.56323  
829.77045  
829.7745  
828.9816  
828.9855  
828.1929  
828.1967  
827.40405  
827.40796  
826.6153  
826.61914  
825.8265  
825.8303  
825.03766  
825.0416  
824.2489  
824.2528  
823.4601  
823.464  
822.67126  
822.6752  
821.8825  
821.8863  
821.0936  
821.0975  
820.30493  
820.3087  
819.516  
819.51996  
818.72723  
818.7311

817.93835  
817.9423  
817.14966  
817.15344  
816.3608  
816.3646  
815.572  
815.576  
814.78314  
814.787  
813.99445  
813.9983  
813.20557  
813.2095  
812.41675  
812.4207  
811.628  
811.6319  
810.8392  
810.84314  
810.05035  
810.05426  
809.2616  
809.26544  
808.4728  
808.4768  
807.684  
807.6879  
806.8952  
806.8991  
806.1064  
806.11017  
805.3176  
805.3214  
804.5288  
804.5326  
803.74  
803.74396  
802.95123  
802.9551  
802.1624  
802.16626  
801.37366  
801.37744  
800.5847

800.5886  
799.79596  
799.79974  
799.00726  
799.0111  
798.21844  
798.2223  
797.42957  
797.4334  
796.64087  
796.6446  
795.85187  
795.8558  
795.0631  
795.0669  
794.27435  
794.27826  
793.4856  
793.4893  
792.6967  
792.70056  
791.908  
791.91174  
791.1191  
791.1229  
790.3304  
790.33417  
789.5415  
789.54535  
788.75275  
788.7566  
787.964  
787.9677  
787.1751  
787.1789  
786.38635  
786.3901  
785.59753  
785.6013  
784.8087  
784.81256  
784.01996  
784.02374  
783.23114  
783.23505

782.4424  
782.44604  
781.65356  
781.6574  
780.86475  
780.86847  
780.076  
780.07965  
779.2872  
779.2909  
778.4983  
778.5021  
777.7096  
777.7132  
776.9208  
776.92456  
776.1319  
776.13574  
775.3432  
775.34686  
774.5543  
774.55804  
773.7656  
773.7692  
772.9768  
772.98047  
772.1879  
772.1916  
771.39923  
771.4029  
770.61035  
770.614  
769.8215  
769.82513  
769.0327  
769.03656  
768.24396  
768.2475  
767.4551  
767.45886  
766.66626  
766.67004  
765.87744  
765.88116  
765.0887

765.09235  
764.29987  
764.3037  
763.5111  
763.5148  
762.7223  
762.72595  
761.9335  
761.9372  
761.1446  
761.1483  
760.3559  
760.3597  
759.5671  
759.57086  
758.7783  
758.7819  
757.9895  
757.99316  
757.2007  
757.20435  
756.4118  
756.4155  
755.6231  
755.6268  
754.8342  
754.8379  
754.0455  
754.049  
753.2567  
753.2603  
752.46796  
752.47156  
751.6791  
751.68274  
750.8902  
750.894  
750.1014  
750.10504  
749.3127  
749.31635  
748.5238  
748.52747  
747.735  
747.73865

746.94617  
746.9498  
746.1574  
746.1611  
745.3687  
745.37213  
744.5798  
744.5835  
743.791  
743.7946  
743.0022  
743.0058  
742.21344  
742.217  
741.4246  
741.42816  
740.6358  
740.63947  
739.84705  
739.85065  
739.0582  
739.06177  
738.2694  
738.2731  
737.4805  
737.48413  
736.69183  
736.6954  
735.9031  
735.90656  
735.11426  
735.1178  
734.32544  
734.329  
733.53656  
733.5401  
732.74786  
732.7513  
731.959  
731.9625  
731.17017  
731.17377  
730.3815  
730.38495  
729.5926

729.5962  
728.8038  
728.80743  
728.015  
728.0185  
727.2262  
727.2297  
726.4374  
726.4409  
725.6486  
725.6521  
724.8598  
724.8633  
724.07104  
724.0745  
723.28217  
723.28577  
722.49335  
722.49695  
721.7046  
721.70807  
720.9158  
720.9193  
720.12695  
720.13043  
719.33813  
719.3417  
718.5494  
718.5529  
717.7606  
717.7641  
716.9718  
716.9752  
716.1829  
716.1865  
715.39417  
715.39764  
714.6054  
714.6089  
713.8165  
713.82  
713.0278  
713.03125  
712.2389  
712.24243

711.45013  
711.4537  
710.6614  
710.6648  
709.8725  
709.87604  
709.08374  
709.0872  
708.2949  
708.2984  
707.50616  
707.5096  
706.71735  
706.7208  
705.9285  
705.932  
705.1397  
705.1432  
704.3509  
704.3544  
703.56213  
703.5656  
702.7734  
702.7768  
701.9845  
701.988  
701.1957  
701.1993  
700.40686  
700.4104  
699.6181  
699.6216  
698.82935  
698.83276  
698.0405  
698.04395  
697.2517  
697.2552  
696.4629  
696.4664  
695.67413  
695.67755  
694.8853  
694.88873  
694.0965

694.1  
693.3077  
693.3111  
692.5188  
692.52234  
691.73  
691.73346  
690.94116  
690.9446  
690.15234  
690.1557  
689.3636  
689.36694  
688.5747  
688.578  
687.7859  
687.78925  
686.997  
687.0004  
686.2082  
686.2116  
685.41943  
685.42285  
684.63055  
684.63403  
683.8417  
683.84515  
683.0529  
683.0563  
682.2641  
682.2675  
681.4753  
681.47864  
680.6864  
680.6898  
679.8976  
679.90094  
679.10876  
679.1122  
678.3199  
678.3233  
677.53107  
677.5345  
676.74225  
676.74567

675.9534  
675.9568  
675.1646  
675.168  
674.37573  
674.3792  
673.5869  
673.59033  
672.7981  
672.8015  
672.0093  
672.0127  
671.22046  
671.2238  
670.43164  
670.43494  
669.64276  
669.64624  
668.85394  
668.85736  
668.0652  
668.0685  
667.2763  
667.2797  
666.4875  
666.4908  
665.69867  
665.70197  
664.90985  
664.9132  
664.121  
664.12427  
663.33215  
663.3355  
662.54333  
662.54663  
661.7545  
661.7579  
660.96564  
660.969  
660.1769  
660.1801  
659.388  
659.39136  
658.5992

658.60254  
657.81036  
657.81366  
657.02155  
657.02484  
656.2327  
656.236  
655.44385  
655.44714  
654.6551  
654.6583  
653.8662  
653.86957  
653.0774  
653.0807  
652.2886  
652.2919  
651.4997  
651.50305  
650.71094  
650.71423  
649.9221  
649.9254  
649.1333  
649.13654  
648.3444  
648.3477  
647.55554  
647.5589  
646.7668  
646.76996  
645.9779  
645.9812  
645.18915  
645.1924  
644.40027  
644.40356  
643.6115  
643.6147  
642.82263  
642.82587  
642.03375  
642.0371  
641.245  
641.2482

640.4561  
640.45935  
639.6673  
639.6706  
638.8785  
638.8817  
638.08966  
638.0929  
637.30084  
637.3041  
636.51196  
636.51526  
635.72314  
635.7264  
634.9343  
634.9375  
634.1455  
634.14874  
633.3567  
633.35986  
632.5678  
632.5711  
631.779  
631.7823  
630.9902  
630.9934  
630.20135  
630.20465  
629.41254  
629.4158  
628.62366  
628.6269  
627.8349  
627.8381  
627.0461  
627.04926  
626.2572  
626.26044  
625.4684  
625.47156  
624.6796  
624.68274  
623.89075  
623.8939  
623.10187

623.1051  
622.3131  
622.3162  
621.52423  
621.52747  
620.73535  
620.7386  
619.9466  
619.94977  
619.1578  
619.1609  
618.36896  
618.3721  
617.58014  
617.5833  
616.7912  
616.7945  
616.00244  
616.0056  
615.2136  
615.2168  
614.4248  
614.428  
613.636  
613.6391  
612.84717  
612.8503  
612.05835  
612.06146  
611.2695  
611.27264  
610.4807  
610.48376  
609.69183  
609.69495  
608.903  
608.9061  
608.11414  
608.11725  
607.3253  
607.32855  
606.53656  
606.5396  
605.7476  
605.7508

604.95886  
604.9619  
604.17004  
604.17316  
603.3812  
603.3843  
602.5924  
602.59546  
601.8035  
601.8066  
601.0148  
601.0178  
600.2259  
600.229  
599.43713  
599.4402  
598.64825  
598.6513  
597.85944  
597.8625  
597.0706  
597.0737  
596.28174  
596.28485  
595.493  
595.496  
594.7041  
594.7072  
593.9153  
593.91833  
593.12646  
593.12946  
592.3376  
592.34064  
591.5487  
591.5518  
590.75995  
590.763  
589.9711  
589.97424  
589.1823  
589.1853  
588.39343  
588.3965  
587.6047

587.60767  
586.8158  
586.8188  
586.027  
586.03  
585.23816  
585.2412  
584.4493  
584.4524  
583.66046  
583.6635  
582.87164  
582.87476  
582.0829  
582.08594  
581.294  
581.29706  
580.5051  
580.5082  
579.7164  
579.7194  
578.9275  
578.93054  
578.13873  
578.1417  
577.34985  
577.3529  
576.56104  
576.5641  
575.7722  
575.7752  
574.9834  
574.9864  
574.1945  
574.1976  
573.4057  
573.4087  
572.6169  
572.6199  
571.82806  
571.83105  
571.03925  
571.0422  
570.2504  
570.2534

569.46155  
569.4646  
568.6727  
568.6757  
567.8839  
567.8869  
567.0951  
567.0981  
566.3063  
566.30927  
565.51746  
565.5204  
564.7286  
564.7316  
563.93976  
563.94275  
563.15094  
563.154  
562.3621  
562.3651  
561.5733  
561.57623  
560.7845  
560.7874  
559.9956  
559.99854  
559.2068  
559.2098  
558.41797  
558.4209  
557.6291  
557.6321  
556.84033  
556.84326  
556.0515  
556.05444  
555.26263  
555.2656  
554.4738  
554.4768  
553.685  
553.6879  
552.8962  
552.89905  
552.10736

552.1103  
551.31854  
551.3214  
550.52966  
550.5326  
549.7409  
549.7438  
548.952  
548.95496  
548.1632  
548.1661  
547.3743  
547.37726  
546.5856  
546.58844  
545.79675  
545.79956  
545.0079  
545.01074  
544.21906  
544.2219  
543.43024  
543.43317  
542.64136  
542.6443  
541.8526  
541.85547  
541.0637  
541.0667  
540.2749  
540.2778  
539.4861  
539.489  
538.69727  
538.70013  
537.90845  
537.9113  
537.1196  
537.1225  
536.33075  
536.3336  
535.54193  
535.5448  
534.7531  
534.75604

533.9643  
533.96716  
533.1755  
533.1783  
532.3866  
532.38947  
531.5978  
531.6006  
530.809  
530.81177  
530.02014  
530.02295  
529.2313  
529.2342  
528.4425  
528.4453  
527.6536  
527.6565  
526.8648  
526.8676  
526.076  
526.0788  
525.2872  
525.2899  
524.49835  
524.5012  
523.7095  
523.71234  
522.9207  
522.9235  
522.1319  
522.1346  
521.343  
521.3458  
520.5542  
520.557  
519.7654  
519.7682  
518.97656  
518.9794  
518.18774  
518.1905  
517.39886  
517.4017  
516.61005

516.61285  
515.8212  
515.824  
515.0324  
515.03516  
514.2435  
514.24634  
513.4548  
513.4575  
512.6659  
512.6687  
511.87708  
511.87985  
511.08823  
511.09103  
510.29944  
510.30225  
509.51056  
509.51337  
508.72177  
508.72458  
507.93295  
507.9357  
507.14417  
507.14685  
506.3553  
506.358  
505.5664  
505.56918  
504.77762  
504.7804  
503.9888  
503.99155  
503.2  
503.20273  
502.41113  
502.41388  
501.62228  
501.6251  
500.8335  
500.83624  
500.04465  
500.04736  
499.25583  
499.25854

498.467  
498.4697  
497.6782  
497.68088  
496.8893  
496.89203  
496.1005  
496.10324  
495.31168  
495.31445  
494.52283  
494.5256  
493.73404  
493.73676  
492.9452  
492.94794  
492.15634  
492.1591  
491.36755  
491.37027  
490.5787  
490.58142  
489.7899  
489.7926  
489.00104  
489.00375  
488.2122  
488.2149  
487.4234  
487.4261  
486.6346  
486.63724  
485.84573  
485.84842  
485.0569  
485.05966  
484.26804  
484.27075  
483.47925  
483.482  
482.6904  
482.6931  
481.90158  
481.90424  
481.1128

481.1154  
480.32388  
480.32666  
479.5351  
479.5378  
478.74625  
478.74896  
477.95746  
477.96014  
477.16864  
477.1713  
476.37982  
476.38248  
475.59094  
475.59363  
474.80215  
474.8048  
474.0133  
474.016  
473.22452  
473.2271  
472.43567  
472.4383  
471.6468  
471.64944  
470.858  
470.86066  
470.0692  
470.07187  
469.2804  
469.28302  
468.49158  
468.49414  
467.70273  
467.70535  
466.91394  
466.91653  
466.12512  
466.1277  
465.3363  
465.33893  
464.5475  
464.5501  
463.75867  
463.76126

462.96988  
462.97244  
462.18106  
462.18362  
461.39218  
461.3948  
460.6034  
460.60602  
459.8146  
459.81717  
459.0258  
459.02838  
458.23697  
458.23953  
457.44815  
457.4507  
456.65936  
456.66193  
455.8705  
455.87308  
455.0817  
455.08423  
454.29285  
454.29544  
453.50406  
453.50665  
452.71527  
452.71783  
451.92645  
451.929  
451.13763  
451.14014  
450.3488  
450.3514  
449.56  
449.56256  
448.77115  
448.77374  
447.98236  
447.9849  
447.1936  
447.1961  
446.40472  
446.40732  
445.61594

445.61847  
444.82712  
444.8296  
444.0383  
444.04077  
443.24945  
443.25195  
442.46066  
442.46317  
441.67184  
441.67435  
440.88306  
440.88553  
440.0942  
440.0967  
439.3054  
439.30795  
438.51657  
438.51907  
437.72778  
437.7303  
436.93896  
436.94144  
436.15015  
436.15262  
435.36133  
435.36386  
434.5725  
434.575  
433.78363  
433.7862  
432.99487  
432.99734  
432.20605  
432.20856  
431.4172  
431.41974  
430.62842  
430.63092  
429.83957  
429.8421  
429.05078  
429.05325  
428.26196  
428.26447

427.47318  
427.4756  
426.68436  
426.68683  
425.89554  
425.898  
425.10672  
425.10916  
424.3179  
424.32037  
423.5291  
423.5316  
422.74026  
422.7427  
421.95145  
421.95392  
421.16263  
421.16507  
420.37384  
420.3763  
419.585  
419.5875  
418.79617  
418.79865  
418.0074  
418.00983  
417.21857  
417.22098  
416.42978  
416.43213  
415.6409  
415.6434  
414.8521  
414.85452  
414.0633  
414.06573  
413.2745  
413.27692  
412.48566  
412.48813  
411.69684  
411.6993  
410.90802  
410.91046  
410.11923

410.12164  
409.3304  
409.3328  
408.54156  
408.54404  
407.75278  
407.75513  
406.96396  
406.96637  
406.17517  
406.17752  
405.38632  
405.38873  
404.5975  
404.59995  
403.8087  
403.81107  
403.0199  
403.02228  
402.23105  
402.23343  
401.44226  
401.44464  
400.65344  
400.65576  
399.86462  
399.867  
399.07584  
399.0782  
398.287  
398.28934  
397.49817  
397.50052  
396.70935  
396.71167  
395.92056  
395.9229  
395.13168  
395.1341  
394.34293  
394.34525  
393.55408  
393.55643  
392.7653  
392.76764

391.9765  
391.97882  
391.18765  
391.19  
390.39886  
390.4012  
389.61002  
389.61234  
388.82123  
388.82352  
388.03244  
388.03473  
387.2436  
387.2459  
386.45474  
386.45706  
385.66592  
385.66827  
384.8771  
384.8795  
384.0883  
384.09064  
383.29947  
383.30182  
382.51068  
382.51303  
381.7219  
381.72418  
380.9331  
380.9353  
380.14423  
380.14655  
379.3554  
379.35773  
378.56662  
378.56888  
377.77777  
377.7801  
376.98898  
376.9913  
376.20013  
376.20245  
375.41135  
375.41367  
374.6225

374.62482  
373.8337  
373.83594  
373.04492  
373.0472  
372.25607  
372.25836  
371.4673  
371.46954  
370.6785  
370.6807  
369.88965  
369.8919  
369.1008  
369.10312  
368.312  
368.31427  
367.52322  
367.52548  
366.73444  
366.73663  
365.94553  
365.9478  
365.15674  
365.15903  
364.36795  
364.37018  
363.5791  
363.58133  
362.79028  
362.79254  
362.0015  
362.00375  
361.21265  
361.21487  
360.42386  
360.4261  
359.63504  
359.63724  
358.84622  
358.84845  
358.05737  
358.05963  
357.26855  
357.27075

356.47974  
356.48193  
355.69092  
355.69315  
354.9021  
354.9043  
354.11325  
354.11548  
353.32446  
353.32666  
352.5356  
352.53784  
351.74683  
351.74902  
350.95798  
350.9602  
350.1692  
350.17136  
349.38034  
349.38257  
348.59152  
348.5937  
347.8027  
347.8049  
347.01385  
347.01608  
346.22507  
346.2272  
345.43625  
345.43842  
344.64743  
344.6496  
343.8586  
343.86078  
343.0698  
343.07196  
342.28098  
342.2831  
341.49213  
341.4943  
340.70328  
340.70544  
339.9145  
339.91666  
339.12564

339.1278  
338.33685  
338.339  
337.548  
337.5502  
336.7592  
336.76132  
335.97037  
335.9725  
335.18158  
335.1837  
334.39276  
334.3949  
333.6039  
333.60605  
332.8151  
332.81723  
332.02628  
332.0284  
331.23743  
331.2396  
330.4486  
330.45074  
329.6598  
329.66193  
328.87097  
328.8731  
328.08215  
328.0843  
327.29333  
327.29544  
326.50452  
326.50662  
325.7157  
325.71777  
324.92685  
324.929  
324.13803  
324.14014  
323.3492  
323.35132  
322.5604  
322.5625  
321.77158  
321.77368

320.98276  
320.98486  
320.19394  
320.196  
319.40512  
319.40723  
318.6163  
318.61838  
317.82748  
317.82956  
317.03864  
317.0407  
316.24982  
316.2519  
315.461  
315.46307  
314.67218  
314.67426  
313.88336  
313.8854  
313.09457  
313.09662  
312.3057  
312.30777  
311.51688  
311.51892  
310.72806  
310.7301  
309.93924  
309.94128  
309.15042  
309.1525  
308.36163  
308.36365  
307.57278  
307.5748  
306.784  
306.786  
305.99515  
305.9972  
305.2063  
305.20834  
304.4175  
304.41956  
303.6287

303.6307  
302.83987  
302.84192  
302.05103  
302.05304  
301.26224  
301.26422  
300.4734  
300.47543  
299.68457  
299.6866  
298.89575  
298.89774  
298.10693  
298.10895  
297.3181  
297.32013  
296.5293  
296.5313  
295.74048  
295.7425  
294.9517  
294.95364  
294.16287  
294.16486  
293.37402  
293.37604  
292.5852  
292.5872  
291.79642  
291.79837  
291.00757  
291.00955  
290.21875  
290.22073  
289.42993  
289.43195  
288.6411  
288.6431  
287.8523  
287.85425  
287.0635  
287.06546  
286.2747  
286.27664

285.48584  
285.48782  
284.69702  
284.699  
283.9082  
283.91016  
283.11935  
283.12134  
282.33057  
282.33255  
281.54175  
281.5437  
280.75293  
280.75485  
279.9641  
279.96603  
279.1753  
279.17722  
278.38647  
278.3884  
277.59766  
277.59958  
276.80884  
276.81076  
276.02002  
276.02194  
275.2312  
275.23315  
274.44238  
274.4443  
273.65356  
273.65546  
272.86478  
272.86664  
272.07596  
272.07785  
271.28714  
271.28903  
270.4983  
270.5002  
269.70947  
269.71136  
268.92065  
268.92255  
268.13184

268.13376  
267.34302  
267.3449  
266.55417  
266.55612  
265.76538  
265.76727  
264.97653  
264.97845  
264.18774  
264.1896  
263.39893  
263.40082  
262.6101  
262.61197  
261.8213  
261.82318  
261.03247  
261.03433  
260.24362  
260.2455  
259.45483  
259.45673  
258.66602  
258.66788  
257.8772  
257.87906  
257.08838  
257.0902  
256.29956  
256.3014  
255.51073  
255.51259  
254.7219  
254.72375  
253.93309  
253.93497  
253.14426  
253.1461  
252.35547  
252.35728  
251.56662  
251.56845  
250.77783  
250.77966

249.98898  
249.99083  
249.20016  
249.202  
248.41135  
248.4132  
247.62253  
247.62436  
246.83371  
246.83554  
246.04488  
246.04674  
245.25609  
245.25789  
244.46724  
244.46906  
243.67842  
243.68027  
242.88963  
242.89145  
242.10081  
242.10262  
241.31197  
241.31378  
240.52318  
240.52496  
239.73433  
239.73613  
238.9455  
238.9473  
238.15671  
238.15851  
237.3679  
237.36966  
236.57907  
236.58087  
235.79027  
235.79202  
235.00143  
235.0032  
234.21259  
234.21439  
233.4238  
233.42557  
232.63496

232.63673  
231.84615  
231.8479  
231.05734  
231.05911  
230.26852  
230.27028  
229.4797  
229.48146  
228.69087  
228.69263  
227.90205  
227.90382  
227.11325  
227.11499  
226.32442  
226.32617  
225.53561  
225.53737  
224.74678  
224.7485  
223.95796  
223.95969  
223.16914  
223.17088  
222.38033  
222.38208  
221.5915  
221.59325  
220.80269  
220.8044  
220.01387  
220.01558  
219.22504  
219.22679  
218.43623  
218.43794  
217.64742  
217.64914  
216.85861  
216.86029  
216.06976  
216.07149  
215.28098  
215.28268

214.49216  
214.49384  
213.70331  
213.70502  
212.91452  
212.91618  
212.1257  
212.1274  
211.33685  
211.33855  
210.54805  
210.54971  
209.75923  
209.76091  
208.97043  
208.97209  
208.1816  
208.18327  
207.39279  
207.39445  
206.60396  
206.60562  
205.81514  
205.81679  
205.02632  
205.028  
204.2375  
204.23917  
203.44868  
203.45035  
202.65985  
202.66153  
201.87105  
201.87271  
201.08221  
201.08386  
200.29341  
200.29506  
199.5046  
199.50623  
198.71577  
198.71744  
197.92694  
197.9286  
197.13812

197.13977  
196.34932  
196.35094  
195.5605  
195.56213  
194.77167  
194.7733  
193.98288  
193.98451  
193.19403  
193.19565  
192.4052  
192.40683  
191.6164  
191.61803  
190.82756  
190.8292  
190.03873  
190.04034  
189.24992  
189.25154  
188.46112  
188.4627  
187.67229  
187.6739  
186.88345  
186.88507  
186.09465  
186.09624  
185.30582  
185.30742  
184.51697  
184.51859  
183.72818  
183.72972  
182.93935  
182.94093  
182.15054  
182.15211  
181.36171  
181.36331  
180.57289  
180.57448  
179.78407  
179.78563

178.99524  
178.9968  
178.2064  
178.20798  
177.4176  
177.41917  
176.62875  
176.63034  
175.83995  
175.84152  
175.05113  
175.05269  
174.2623  
174.26385  
173.47348  
173.47505  
172.68466  
172.6862  
171.89583  
171.8974  
171.10701  
171.10857  
170.31819  
170.31975  
169.52937  
169.53091  
168.74055  
168.74208  
167.95174  
167.95326  
167.1629  
167.16446  
166.3741  
166.37563  
165.58525  
165.58678  
164.79645  
164.79797  
164.00763  
164.00916  
163.2188  
163.22034  
162.42998  
162.43149  
161.64114

161.64267  
160.85234  
160.85385  
160.0635  
160.065  
159.27469  
159.27618  
158.48587  
158.48737  
157.69705  
157.69855  
156.90822  
156.90973  
156.1194  
156.1209  
155.33058  
155.33206  
154.54175  
154.54326  
153.75294  
153.75441  
152.96411  
152.9656  
152.1753  
152.17677  
151.38646  
151.38795  
150.59764  
150.59912  
149.80882  
149.8103  
149.02  
149.02147  
148.23117  
148.23265  
147.44235  
147.44382  
146.65352  
146.655  
145.86472  
145.86617  
145.07588  
145.07735  
144.28706  
144.28853

143.49825  
143.4997  
142.70943  
142.71086  
141.9206  
141.92206  
141.13177  
141.13321  
140.34296  
140.3444  
139.55414  
139.55557  
138.76532  
138.76675  
137.9765  
137.97792  
137.18767  
137.18909  
136.39886  
136.40028  
135.61005  
135.61147  
134.82123  
134.82266  
134.0324  
134.03383  
133.24359  
133.24501  
132.45477  
132.45619  
131.66595  
131.66737  
130.87714  
130.87856  
130.08833  
130.08972  
129.29953  
129.30092  
128.5107  
128.5121  
127.72189  
127.72328  
126.933075  
126.93445  
126.14426

126.14565  
125.35544  
125.35683  
124.56662  
124.56801  
123.7778  
123.77919  
122.989  
122.99038  
122.20017  
122.20156  
121.411354  
121.41273  
120.62256  
120.62391  
119.833725  
119.8351  
119.04492  
119.04628  
118.2561  
118.25747  
117.467285  
117.46864  
116.67847  
116.67983  
115.88965  
115.89101  
115.100845  
115.10219  
114.31203  
114.31338  
113.52321  
113.52456  
112.7344  
112.735725  
111.94559  
111.94693  
111.15692  
111.15816  
110.368385  
110.36948  
109.57988  
109.581  
108.79155  
108.792656

108.00364  
108.00494  
107.21618  
107.21757  
106.42904  
106.43029  
105.64279  
105.64369  
104.8578  
104.85853  
104.07375  
104.07511  
103.29118  
103.293465  
102.51156  
102.51511  
101.7357  
101.73974  
100.96394  
100.968056  
100.19611  
100.200645  
99.43264  
99.438484  
98.67322  
98.68061  
97.9189  
97.92828  
97.16813  
97.18083  
96.421646  
96.43876  
95.67953  
95.70252  
94.94149  
94.9717  
94.20876  
94.24697  
93.480034  
93.52546  
92.75569  
92.808556  
92.03506  
92.095924  
91.31854

91.38684  
90.60755  
90.68377  
89.90215  
89.98529  
89.20221  
89.291885  
88.50686  
88.60336  
87.816765  
87.91971  
87.13176  
87.24081  
86.4518  
86.566986  
85.77669  
85.89698  
85.10687  
85.23311  
84.44314  
84.57549  
83.78607  
83.924644  
83.13564  
83.27974  
82.49277  
82.64173  
81.85713  
82.010155  
81.22819  
81.384674  
80.605545  
80.76703  
79.98976  
80.15565  
79.379906  
79.55014  
78.77701  
78.95131  
78.18198  
78.36113  
77.59319  
77.77769  
77.010826  
77.199524

76.43567  
76.62739  
75.86611  
76.060776  
75.30352  
75.500374  
74.74626  
74.9445  
74.19414  
74.393364  
73.64597  
73.84709  
73.10191  
73.30431  
72.56237  
72.76671  
72.026344  
72.23352  
71.493385  
71.70372  
70.96444  
71.17768  
70.440384  
70.655396  
69.92038  
70.136925  
69.40348  
69.62064  
68.89077  
69.108505  
68.381485  
68.59986  
67.87542  
68.09403  
67.37252  
67.59046  
66.872406  
67.08922  
66.374756  
66.59036  
65.879425  
66.0938  
65.38633  
65.59917  
64.89466

65.10622  
64.40454  
64.61537  
63.916122  
64.126045  
63.42996  
63.638306  
62.94651  
63.153503  
62.46512  
62.67079  
61.985798  
62.189323  
61.508297  
61.70951  
61.03265  
61.23166  
60.55863  
60.755917  
60.086105  
60.281227  
59.614777  
59.808094  
59.144463  
59.336388  
58.6755  
58.86604  
58.208424  
58.397533  
57.743046  
57.931046  
57.278557  
57.46649  
56.815228  
57.00371  
56.35351  
56.54225  
55.893238  
56.082302  
55.43489  
55.624252  
54.978245  
55.167908  
54.52335  
54.714252

54.069973  
54.262173  
53.617817  
53.812164  
53.167862  
53.36398  
52.72101  
52.91775  
52.27702  
52.473488  
51.836075  
52.031696  
51.397705  
51.592384  
50.962368  
51.1564  
50.52944  
50.72257  
50.099617  
50.29145  
49.67245  
49.862946  
49.24898  
49.438477  
48.828873  
49.017147  
48.4124  
48.598896  
47.999584  
48.184692  
47.590027  
47.7736  
47.18453  
47.365494  
46.783424  
46.961475  
46.385532  
46.561386  
45.99194  
46.16523  
45.602894  
45.773808  
45.21826  
45.385914  
44.838997

45.003212  
44.46506  
44.62604  
44.094986  
44.25246  
43.729324  
43.883915  
43.36838  
43.519722  
43.012497  
43.160423  
42.660072  
42.80495  
42.311974  
42.453796  
41.96784  
42.106327  
41.62928  
41.764256  
41.29567  
41.426895  
40.967937  
41.095398  
40.64572  
40.77027  
40.32908  
40.45012  
40.018562  
40.13599  
39.71287  
39.82705  
39.41202  
39.52356  
39.11554  
39.224506  
38.823994  
38.930225  
38.53807  
38.641644  
38.25804  
38.359257  
37.98361  
38.082912  
37.71495  
37.812504

37.451447  
37.548595  
37.19366  
37.29138  
36.94221  
37.04078  
36.696583  
36.796577  
36.45622  
36.55916  
36.220676  
36.326782  
35.991077  
36.099598  
35.768234  
35.87852  
35.550938  
35.662853  
35.339035  
35.45184  
35.13183  
35.246384  
34.929306  
35.04663  
34.73184  
34.85231  
34.538998  
34.66301  
34.351845  
34.47944  
34.169914  
34.30085  
33.992874  
34.126297  
33.82077  
33.956806  
33.652992  
33.791977  
33.48927  
33.631077  
33.33069  
33.474857  
33.1767  
33.32326  
33.027077

33.175877  
32.88099  
33.032238  
32.739292  
32.892597  
32.602383  
32.756958  
32.46932  
32.6255  
32.340004  
32.498253  
32.214066  
32.37432  
32.09212  
32.25449  
31.973583  
32.138046  
31.858288  
32.024704  
31.746296  
31.913937  
31.637758  
31.80594  
31.533054  
31.70111  
31.431854  
31.599857  
31.334133  
31.502193  
31.239426  
31.407227  
31.147894  
31.315424  
31.058857  
31.226507  
30.97232  
31.139944  
30.889277  
31.056557  
30.808937  
30.97594  
30.73167  
30.89787  
30.656963  
30.82248

30.584545  
30.749792  
30.514643  
30.679914  
30.447193  
30.612597  
30.38189  
30.547733  
30.318861  
30.484764  
30.257952  
30.423988  
30.198936  
30.365118  
30.141775  
30.308037  
30.086632  
30.252747  
30.03321  
30.19953  
29.981436  
30.148478  
29.93156  
30.099575  
29.883461  
30.05242  
29.837044  
30.007025  
29.79215  
29.963326  
29.748655  
29.921144  
29.706434  
29.880056  
29.665773  
29.840195  
29.626501  
29.80148  
29.58861  
29.764051  
29.551977  
29.72798  
29.516764  
29.69331  
29.482803

29.659744  
29.449892  
29.627153  
29.418133  
29.595602  
29.38766  
29.56522  
29.358334  
29.53597  
29.330112  
29.507748  
29.302893  
29.480488  
29.276796  
29.454422  
29.251637  
29.429455  
29.227247  
29.405422  
29.203749  
29.382324  
29.181255  
29.36032  
29.15934  
29.33903  
29.138268  
29.318417  
29.117992  
29.298529  
29.098507  
29.27941  
29.079765  
29.2611  
29.06167  
29.24366  
29.044193  
29.227018  
29.027374  
29.211025  
29.011213  
29.195667  
28.995712  
29.18078  
28.980795  
29.166386

28.96637  
29.15275  
28.952557  
29.13972  
28.939383  
29.127188  
28.926744  
29.115088  
28.914726  
29.103605  
28.903193  
29.092665  
28.892052  
29.082127  
28.881266  
29.0719  
28.870739  
29.061918  
28.860704  
29.05237  
28.851088  
29.043173  
28.841917  
29.034334  
28.833038  
29.025755  
28.824497  
29.017498  
28.816267  
29.009518  
28.808365  
29.001862  
28.800833  
28.994564  
28.79372  
28.987543  
28.786926  
28.980806  
28.78042  
28.97425  
28.774157  
28.967953  
28.768091  
28.961935  
28.762241

28.956123  
28.756628  
28.950506  
28.751247  
28.945139  
28.746067  
28.940039  
28.741098  
28.935131  
28.736294  
28.930422  
28.731691  
28.925886  
28.727339  
28.921627  
28.723166  
28.917543  
28.71914  
28.913574  
28.7153  
28.909792  
28.711573  
28.906143  
28.708012  
28.902601  
28.704641  
28.899242  
28.701405  
28.896029  
28.698267  
28.892914  
28.695286  
28.889961  
28.692425  
28.887157  
28.68969  
28.884441  
28.68712  
28.881857  
28.684662  
28.87937  
28.682318  
28.87701  
28.680056  
28.874733

28.677834  
28.872515  
28.675707  
28.870354  
28.673681  
28.868305  
28.671736  
28.866358  
28.669874  
28.864511  
28.668077  
28.862736  
28.666346  
28.861042  
28.66467  
28.859413  
28.66308  
28.857874  
28.661562  
28.856398  
28.660093  
28.854969  
28.658669  
28.853584  
28.657282  
28.852226  
28.655952  
28.850899  
28.654661  
28.849628  
28.653406  
28.848425  
28.65219  
28.847265  
28.65101  
28.846178  
28.64987  
28.84517  
28.648758  
28.84421  
28.647694  
28.843307  
28.646675  
28.842457  
28.645702

28.841637  
28.644783  
28.840849  
28.643896  
28.840096  
28.643044  
28.839376  
28.642235  
28.838694  
28.641449  
28.838032  
28.6407  
28.837399  
28.639992  
28.836802  
28.639315  
28.836239  
28.638659  
28.83569  
28.638037  
28.83518  
28.637442  
28.834703  
28.636875  
28.834259  
28.636328  
28.833843  
28.6358  
28.833443  
28.63529  
28.833044  
28.634804  
28.832659  
28.634333  
28.83229  
28.63388  
28.831934  
28.633455  
28.831594  
28.633041  
28.831278  
28.632643  
28.830988  
28.63225  
28.830711

28.63186  
28.830446  
28.631487  
28.830194  
28.631119  
28.82995  
28.630764  
28.829714  
28.63042  
28.829485  
28.630081  
28.829254  
28.629757  
28.829027  
28.629456  
28.82882  
28.629162  
28.828615  
28.628881  
28.828424  
28.628613  
28.828247  
28.628355  
28.828074  
28.628105  
28.827917  
28.62787  
28.827763  
28.627636  
28.827614  
28.627407  
28.827467  
28.627182  
28.827324  
28.626965  
28.827187  
28.626762  
28.827063  
28.626566  
28.82694  
28.626373  
28.826818  
28.626192  
28.826694  
28.626005

28.826578  
28.62583  
28.826464  
28.625664  
28.826365  
28.625498  
28.826263  
28.625343  
28.826168  
28.625193  
28.826078  
28.62505  
28.82599  
28.624914  
28.825903  
28.624777  
28.825823  
28.624653  
28.825747  
28.624535  
28.82568  
28.624414  
28.825613  
28.6243  
28.825548  
28.624195  
28.825489  
28.6241  
28.825441  
28.623997  
28.825386  
28.6239  
28.825335  
28.623808  
28.825293  
28.623716  
28.825245  
28.623623  
28.825193  
28.623537  
28.825151  
28.623455  
28.825106  
28.62337  
28.825064

28.623285  
28.825024  
28.623203  
28.824986  
28.623125  
28.824947  
28.623047  
28.824907  
28.622967  
28.824877  
28.622896  
28.824839  
28.622828  
28.824806  
28.622765  
28.824778  
28.6227  
28.824738  
28.622643  
28.824717  
28.62258  
28.824684  
28.622519  
28.824656  
28.622458  
28.824625  
28.622398  
28.824598  
28.622345  
28.824566  
28.62229  
28.824545  
28.622236  
28.824522  
28.622187  
28.824497  
28.622135  
28.824476  
28.622084  
28.824451  
28.622036  
28.824434  
28.621988  
28.824411  
28.621943

28.82439  
28.621895  
28.82437  
28.621851  
28.824345  
28.621807  
28.824327  
28.621765  
28.824305  
28.621717  
28.824284  
28.621677  
28.824265  
28.621634  
28.824242  
28.621595  
28.82422  
28.62156  
28.824203  
28.621517  
28.824183  
28.621477  
28.824156  
28.621439  
28.824135  
28.6214  
28.824116  
28.621368  
28.824097  
28.621334  
28.82408  
28.621298  
28.82406  
28.621264  
28.82404  
28.621233  
28.824017  
28.621202  
28.824001  
28.621168  
28.823986  
28.621138  
28.823963  
28.621103  
28.823948

28.62107  
28.823933  
28.62104  
28.823915  
28.621012  
28.8239  
28.620983  
28.823883  
28.62095  
28.823862  
28.620924  
28.823847  
28.620892  
28.823828  
28.620865  
28.823809  
28.620838  
28.823793  
28.62081  
28.823778  
28.620777  
28.823761  
28.620756  
28.823746  
28.62073  
28.823729  
28.620703  
28.823711  
28.620674  
28.823692  
28.620651  
28.82368  
28.620626  
28.823662  
28.6206  
28.823648  
28.620571  
28.823633  
28.620548  
28.823622  
28.62052  
28.8236  
28.620497  
28.823587  
28.620468

28.823574  
28.620445  
28.823559  
28.62042  
28.82354  
28.620398  
28.823526  
28.62037  
28.823505  
28.62035  
28.823494  
28.620325  
28.82348  
28.6203  
28.823463  
28.620274  
28.82345  
28.620253  
28.823433  
28.620226  
28.82342  
28.620203  
28.823404  
28.620178  
28.823387  
28.620155  
28.823372  
28.620132  
28.823357  
28.620106  
28.82334  
28.620079  
28.823324  
28.620062  
28.823307  
28.620035  
28.823294  
28.620014  
28.823277  
28.619991  
28.82326  
28.61997  
28.823244  
28.619946  
28.823227

28.61992  
28.823212  
28.619898  
28.82319  
28.619879  
28.82318  
28.619854  
28.823164  
28.61983  
28.823143  
28.619808  
28.823126  
28.61979  
28.82311  
28.619764  
28.823095  
28.619743  
28.823078  
28.619719  
28.823057  
28.6197  
28.823046  
28.619675  
28.823023  
28.619656  
28.82301  
28.619635  
28.82299  
28.61961  
28.822971  
28.619585  
28.82295  
28.619564  
28.822935  
28.619545  
28.822918  
28.61952  
28.822891  
28.619505  
28.822878  
28.619476  
28.82286  
28.619455  
28.822838  
28.619434

28.822817  
28.619417  
28.822802  
28.619396  
28.822784  
28.61937  
28.822762  
28.619352  
28.822742  
28.619331  
28.822723  
28.619308  
28.822704  
28.619287  
28.822683  
28.619267  
28.822668  
28.619244  
28.822647  
28.619223  
28.822624  
28.6192  
28.822607  
28.619179  
28.822582  
28.619158  
28.822561  
28.619135  
28.822542  
28.619114  
28.822527  
28.619095  
28.822506  
28.619074  
28.822487  
28.61905  
28.822466  
28.619028  
28.822443  
28.61901  
28.822426  
28.61899  
28.822407  
28.61897  
28.822386

28.618946  
28.822367  
28.618927  
28.822348  
28.618902  
28.822327  
28.618883  
28.822302  
28.618862  
28.82228  
28.618843  
28.822262  
28.618818  
28.822247  
28.618797  
28.82222  
28.618774  
28.822203  
28.618755  
28.822182  
28.618734  
28.822165  
28.618713  
28.82214  
28.618694  
28.82212  
28.618675  
28.822102  
28.618649  
28.82208  
28.618626  
28.82206  
28.61861  
28.822039  
28.618587  
28.822018  
28.618565  
28.822  
28.618546  
28.82198  
28.618523  
28.82196  
28.618502  
28.821941  
28.61848

28.82192  
28.618462  
28.8219  
28.618443  
28.82188  
28.61842  
28.821861  
28.618397  
28.821838  
28.618374  
28.821817  
28.618355  
28.821798  
28.618334  
28.821777  
28.61831  
28.821756  
28.61829  
28.821737  
28.618269  
28.821716  
28.61825  
28.821697  
28.618225  
28.821676  
28.618206  
28.821653  
28.618185  
28.821636  
28.618162  
28.821611  
28.61814  
28.821592  
28.61812  
28.821573  
28.618101  
28.821556  
28.618078  
28.821533  
28.618057  
28.82151  
28.618036  
28.82149  
28.61802  
28.821472

28.617998  
28.821451  
28.61797  
28.82143  
28.61795  
28.82141  
28.61793  
28.821388  
28.61791  
28.821365  
28.61789  
28.821344  
28.617865  
28.821321  
28.617846  
28.821302  
28.617825  
28.821283  
28.617802  
28.821259  
28.61778  
28.821238  
28.617762  
28.821217  
28.617743  
28.821196  
28.617716  
28.821175  
28.617697  
28.821154  
28.617678  
28.821135  
28.617657  
28.821114  
28.617634  
28.821089  
28.617613  
28.821068  
28.617594  
28.82105  
28.617573  
28.82103  
28.61755  
28.821007  
28.61753

28.820984  
28.61751  
28.820967  
28.617489  
28.820944  
28.617466  
28.820923  
28.617445  
28.8209  
28.617426  
28.820877  
28.617401  
28.820856  
28.617382  
28.820835  
28.617357  
28.820814  
28.617336  
28.820795  
28.617317  
28.820774  
28.617296  
28.82075  
28.617273  
28.820728  
28.617254  
28.820707  
28.617233  
28.820688  
28.617212  
28.820667  
28.61719  
28.820646  
28.61717  
28.820625  
28.617151  
28.820602  
28.617125  
28.820581  
28.61711  
28.82056  
28.617086  
28.82054  
28.617065  
28.820518

28.617044  
28.820496  
28.617022  
28.820475  
28.617  
28.820457  
28.616978  
28.820433  
28.616957  
28.820414  
28.616936  
28.820393  
28.616917  
28.820374  
28.616898  
28.820347  
28.616873  
28.820326  
28.616854  
28.82031  
28.616835  
28.820286  
28.61681  
28.820265  
28.616789  
28.820246  
28.616768  
28.82022  
28.61675  
28.820204  
28.616724  
28.820179  
28.616701  
28.82016  
28.61668  
28.820139  
28.616665  
28.820118  
28.616642  
28.820097  
28.616617  
28.820072  
28.616596  
28.820051  
28.616577

28.820032  
28.616556  
28.820011  
28.616537  
28.81999  
28.616512  
28.81997  
28.616497  
28.819946  
28.616474  
28.819925  
28.616447  
28.819904  
28.61643  
28.819883  
28.61641  
28.819862  
28.616388  
28.81984  
28.616365  
28.819818  
28.616344  
28.819798  
28.616323  
28.819777  
28.6163  
28.819757  
28.616282  
28.819735  
28.616257  
28.819712  
28.616241  
28.819695  
28.616217  
28.81967  
28.616196  
28.819649  
28.616177  
28.819632  
28.616154  
28.81961  
28.616133  
28.819584  
28.616112  
28.819569

28.616089  
28.819544  
28.616068  
28.819523  
28.616053  
28.819502  
28.616028  
28.819483  
28.616005  
28.819456  
28.615986  
28.819435  
28.615965  
28.819416  
28.615944  
28.819395  
28.615921  
28.819376  
28.615906  
28.819355  
28.615881  
28.819334  
28.615856  
28.819307  
28.615837  
28.819286  
28.615818  
28.81927  
28.615797  
28.819248  
28.615776  
28.81923  
28.615751  
28.8192  
28.615734  
28.819183  
28.61571  
28.819162  
28.615688  
28.819141  
28.61567  
28.81912  
28.615648  
28.8191  
28.615623

28.819078  
28.615604  
28.819056  
28.61558  
28.819029  
28.615564  
28.819014  
28.615541  
28.818989  
28.61552  
28.818968  
28.615501  
28.818949  
28.61548  
28.818928  
28.615456  
28.818907  
28.615435  
28.818886  
28.615416  
28.818865  
28.615393  
28.818842  
28.615372  
28.81882  
28.615353  
28.818806  
28.615332  
28.818785  
28.615309  
28.818764  
28.615288  
28.818739  
28.615265  
28.818714  
28.615244  
28.818697  
28.615229  
28.818678  
28.615204  
28.818657  
28.615181  
28.818632  
28.615158  
28.818607

28.61514  
28.81859  
28.61512  
28.818565  
28.615097  
28.818546  
28.615076  
28.818525  
28.615057  
28.818504  
28.615032  
28.818485  
28.615011  
28.81846  
28.61499  
28.818443  
28.614971  
28.818422  
28.614946  
28.8184  
28.614927  
28.818378  
28.614908  
28.818356  
28.61489  
28.818336  
28.614864  
28.818316  
28.614843  
28.818293  
28.61482  
28.818272  
28.614805  
28.818253  
28.614784  
28.81823  
28.614756  
28.818209  
28.614735  
28.818184  
28.614716  
28.818165  
28.614695  
28.818142  
28.614676

28.818125  
28.61465  
28.818102  
28.614632  
28.818085  
28.614613  
28.818064  
28.614588  
28.81804  
28.614573  
28.818022  
28.61455  
28.818  
28.614527  
28.817978  
28.614508  
28.817951  
28.614485  
28.81793  
28.614464  
28.81791  
28.614443  
28.817888  
28.61442  
28.81787  
28.6144  
28.817848  
28.614376  
28.817823  
28.614355  
28.817802  
28.614334  
28.817783  
28.614317  
28.817764  
28.614292  
28.817743  
28.61427  
28.817717  
28.614252  
28.817696  
28.614237  
28.817686  
28.614206  
28.817656

28.614187  
28.817636  
28.614166  
28.817614  
28.614143  
28.817589  
28.614128  
28.817574  
28.614103  
28.817553  
28.614082  
28.81753  
28.614058  
28.817509  
28.61404  
28.817488  
28.61402  
28.817467  
28.613998  
28.817446  
28.613976  
28.817421  
28.613955  
28.817402  
28.613937  
28.817385  
28.61391  
28.81736  
28.613895  
28.817339  
28.613873  
28.817318  
28.613848  
28.817291  
28.613829  
28.817274  
28.613808  
28.817259  
28.613787  
28.817232  
28.61376  
28.817207  
28.613743  
28.817188  
28.613722

28.817167  
28.613699  
28.817148  
28.613678  
28.817125  
28.613657  
28.817106  
28.613638  
28.817083  
28.613619  
28.81706  
28.613596  
28.81704  
28.613575  
28.817019  
28.613554  
28.816998  
28.613531  
28.816978  
28.61351  
28.81696  
28.613491  
28.816938  
28.613468  
28.816917  
28.613451  
28.816896  
28.613428  
28.816874  
28.613403  
28.816853  
28.613384  
28.816837  
28.613363  
28.81681  
28.613342  
28.816784  
28.61332  
28.816767  
28.6133  
28.816746  
28.61328  
28.816725  
28.613255  
28.816702

28.61324  
28.816683  
28.613214  
28.816658  
28.613194  
28.816639  
28.613174  
28.816618  
28.613152  
28.816597  
28.613132  
28.816576  
28.61311  
28.816555  
28.61309  
28.816532  
28.613068  
28.816515  
28.613047  
28.81649  
28.613024  
28.816471  
28.613003  
28.81645  
28.612984  
28.816425  
28.612963  
28.816408  
28.61294  
28.816383  
28.612919  
28.816362  
28.612898  
28.816343  
28.612875  
28.81632  
28.612856  
28.816301  
28.612833  
28.816277  
28.612814  
28.816256  
28.612791  
28.816235  
28.612766

28.816214  
28.612751  
28.81619  
28.61273  
28.81617  
28.612707  
28.816149  
28.612686  
28.816128  
28.612665  
28.816109  
28.612642  
28.816084  
28.612621  
28.816069  
28.612598  
28.816042  
28.612577  
28.816021  
28.612562  
28.816002  
28.61254  
28.81598  
28.612514  
28.81596  
28.612495  
28.815939  
28.612474  
28.815914  
28.612453  
28.815893  
28.612434  
28.815876  
28.61241  
28.815853  
28.61239  
28.815832  
28.61237  
28.815813  
28.61235  
28.815792  
28.612322  
28.81577  
28.612307  
28.815748

28.612286  
28.815727  
28.612263  
28.815704  
28.612242  
28.815685  
28.612219  
28.815664  
28.612202  
28.815641  
28.612177  
28.81562  
28.61216  
28.8156  
28.612139  
28.815578  
28.612116  
28.815557  
28.612095  
28.815538  
28.612074  
28.815517  
28.61205  
28.815493  
28.61203  
28.815477  
28.61201  
28.81545  
28.611986  
28.815426  
28.611965  
28.815407  
28.611946  
28.815386  
28.611927  
28.815367  
28.6119  
28.815344  
28.611881  
28.815327  
28.61186  
28.815304  
28.611837  
28.815279  
28.611818

28.815258  
28.611797  
28.815237  
28.611774  
28.815216  
28.611753  
28.815193  
28.611734  
28.815178  
28.611715  
28.815151  
28.611694  
28.81513  
28.611671  
28.815113  
28.61165  
28.815086  
28.61163  
28.815065  
28.611607  
28.815048  
28.611588  
28.815023  
28.611565  
28.815002  
28.611542  
28.814981  
28.611526  
28.814964  
28.611504  
28.814943  
28.611483  
28.814917  
28.611462  
28.814896  
28.611439  
28.814878  
28.611418  
28.814857  
28.611397  
28.814837  
28.611374  
28.814816  
28.611355  
28.814795

28.61133  
28.814772  
28.61131  
28.81475  
28.61129  
28.81473  
28.61127  
28.814709  
28.611246  
28.814688  
28.611225  
28.814667  
28.611206  
28.814644  
28.611185  
28.814623  
28.611162  
28.814602  
28.611141  
28.81458  
28.611122  
28.814556  
28.611103  
28.814537  
28.611082  
28.814516  
28.61106  
28.814495  
28.611038  
28.814474  
28.611017  
28.814457  
28.610994  
28.814434  
28.610973  
28.81441  
28.610949  
28.814394  
28.61093  
28.814373  
28.610909  
28.814346  
28.610886  
28.814327  
28.610865

28.814302  
28.610846  
28.814281  
28.610826  
28.81426  
28.6108  
28.81424  
28.610783  
28.814215  
28.610762  
28.814196  
28.61074  
28.814175  
28.610718  
28.814154  
28.610699  
28.814138  
28.61068  
28.814116  
28.610657  
28.814095  
28.610634  
28.814074  
28.610611  
28.814047  
28.610594  
28.814032  
28.610577  
28.814013  
28.61055  
28.813986  
28.61053  
28.813967  
28.610508  
28.813946  
28.610489  
28.813925  
28.61047  
28.813904  
28.610445  
28.81388  
28.61042  
28.81386  
28.610405  
28.813839

28.61038  
28.813818  
28.610355  
28.813793  
28.610336  
28.813774  
28.610315  
28.813751  
28.610292  
28.81373  
28.610273  
28.813707  
28.610254  
28.81369  
28.610235  
28.81367  
28.610212  
28.813646  
28.61019  
28.81362  
28.61017  
28.813604  
28.61015  
28.813583  
28.610128  
28.813562  
28.610106  
28.81354  
28.610086  
28.813522  
28.610064  
28.813496  
28.61004  
28.813477  
28.61002  
28.813456  
28.610003  
28.813437  
28.609982  
28.813412  
28.609957  
28.81339  
28.609938  
28.81337  
28.609913

28.813349  
28.609894  
28.81333  
28.60987  
28.813305  
28.609852  
28.813288  
28.609833  
28.813265  
28.609814  
28.813248  
28.609787  
28.813225  
28.60977  
28.813204  
28.609749  
28.813185  
28.609726  
28.813162  
28.609705  
28.81314  
28.609686  
28.813118  
28.609661  
28.813097  
28.609642  
28.813076  
28.609621  
28.813051  
28.609602  
28.813034  
28.609581  
28.813011  
28.609556  
28.812986  
28.609531  
28.812967  
28.609516  
28.812948  
28.609493  
28.812927  
28.609472  
28.812906  
28.60945  
28.812883

28.60943  
28.812862  
28.609407  
28.812841  
28.60939  
28.81282  
28.60937  
28.812803  
28.609344  
28.812775  
28.609325  
28.81276  
28.609304  
28.812735  
28.609282  
28.812714  
28.60926  
28.812695  
28.60924  
28.81267  
28.609217  
28.812649  
28.609196  
28.812632  
28.609175  
28.812607  
28.609156  
28.812586  
28.609137  
28.812567  
28.609108  
28.812542  
28.609093  
28.812527  
28.609072  
28.8125  
28.609047  
28.812479  
28.609028  
28.812462  
28.60901  
28.81244  
28.608984  
28.81242  
28.608963

28.812399  
28.608944  
28.812378  
28.608925  
28.812355  
28.6089  
28.812332  
28.608881  
28.812313  
28.60886  
28.812292  
28.608837  
28.812267  
28.608816  
28.812246  
28.608795  
28.812227  
28.608776  
28.812206  
28.608751  
28.81218  
28.608728  
28.812164  
28.608713  
28.812145  
28.608688  
28.81212  
28.60867  
28.8121  
28.608648  
28.812078  
28.608627  
28.812057  
28.608604  
28.812037  
28.608585  
28.812014  
28.60856  
28.811993  
28.60854  
28.811972  
28.60852  
28.81195  
28.6085  
28.81193

28.60848  
28.811909  
28.608458  
28.811886  
28.608437  
28.811865  
28.608416  
28.811846  
28.60839  
28.811823  
28.608372  
28.811802  
28.60835  
28.811783  
28.608332  
28.811758  
28.608313  
28.811743  
28.60829  
28.811718  
28.608267  
28.811695  
28.608244  
28.811672  
28.608223  
28.811651  
28.608204  
28.81163  
28.608185  
28.811615  
28.60816  
28.811588  
28.608145  
28.811571  
28.608116  
28.811544  
28.608095  
28.811527  
28.608082  
28.811508  
28.608055  
28.811481  
28.608036  
28.811462  
28.608011

28.811438  
28.607994  
28.811422  
28.607971  
28.811401  
28.607948  
28.811377  
28.607927  
28.811356  
28.607906  
28.811337  
28.607887  
28.811316  
28.607868  
28.811295  
28.607843  
28.811274  
28.607824  
28.811253  
28.607803  
28.811234  
28.60778  
28.811213  
28.60776  
28.811188  
28.607737  
28.811167  
28.607716  
28.811146  
28.607695  
28.811123  
28.607676  
28.811102  
28.607653  
28.81108  
28.607634  
28.81106  
28.607613  
28.81104  
28.607592  
28.811016  
28.607569  
28.810999  
28.607548  
28.81098

28.607529  
28.810959  
28.607504  
28.810934  
28.607483  
28.810913  
28.607466  
28.810894  
28.607439  
28.810867  
28.607418  
28.810848  
28.6074  
28.810831  
28.607378  
28.810804  
28.607355  
28.810787  
28.607334  
28.81076  
28.607315  
28.810745  
28.607292  
28.81072  
28.607271  
28.8107  
28.60725  
28.810675  
28.607233  
28.81066  
28.607208  
28.810635  
28.60719  
28.810617  
28.607166  
28.810596  
28.607147  
28.810574  
28.607124  
28.810553  
28.607103  
28.810526  
28.607082  
28.81051  
28.607063

28.81049  
28.607038  
28.810467  
28.60702  
28.810446  
28.606995  
28.810423  
28.60698  
28.810404  
28.60695  
28.810379  
28.606937  
28.810364  
28.606916  
28.810339  
28.60689  
28.810318  
28.60687  
28.810297  
28.606848  
28.810276  
28.606827  
28.810251  
28.606808  
28.810232  
28.606787  
28.810211  
28.606768  
28.81019  
28.606745  
28.810171  
28.606724  
28.810152  
28.606703  
28.81013  
28.60668  
28.810104  
28.606659  
28.810085  
28.60664  
28.810062  
28.606615  
28.810041  
28.606594  
28.810019

28.606573  
28.809998  
28.606556  
28.809977  
28.606535  
28.809958  
28.606512  
28.809937  
28.606491  
28.809917  
28.60647  
28.809896  
28.606447  
28.809875  
28.606426  
28.809855  
28.606403  
28.809828  
28.606384  
28.809809  
28.606361  
28.809784  
28.606339  
28.809763  
28.606318  
28.809744  
28.606298  
28.809721  
28.60628  
28.809704  
28.606258  
28.80968  
28.606234  
28.809656  
28.606216  
28.809643  
28.606194  
28.809616  
28.60617  
28.809599  
28.60615  
28.809574  
28.60613  
28.809553  
28.60611

28.809534  
28.60609  
28.809513  
28.606068  
28.809488  
28.606047  
28.80947  
28.606026  
28.809446  
28.606007  
28.80943  
28.605982  
28.809406  
28.605965  
28.809385  
28.60594  
28.809362  
28.605919  
28.809341  
28.6059  
28.80932  
28.605879  
28.809301  
28.605856  
28.809278  
28.605835  
28.809258  
28.605814  
28.809235  
28.605791  
28.809214  
28.60577  
28.809193  
28.60575  
28.809172  
28.605726  
28.80915  
28.605707  
28.809132  
28.605688  
28.809107  
28.605665  
28.809086  
28.60564  
28.809065

28.605623  
28.809048  
28.605602  
28.809021  
28.605581  
28.809  
28.605558  
28.808979  
28.605543  
28.80896  
28.605515  
28.808937  
28.605495  
28.808916  
28.605474  
28.808893  
28.605455  
28.808872  
28.605434  
28.808853  
28.605412  
28.808832  
28.60539  
28.808813  
28.60537  
28.80879  
28.605347  
28.808765  
28.605326  
28.808746  
28.605307  
28.80873  
28.605286  
28.808704  
28.605263  
28.808685  
28.605242  
28.808664  
28.605225  
28.808645  
28.605202  
28.808622  
28.605179  
28.808598  
28.605158

28.808577  
28.605139  
28.808558  
28.605118  
28.808537  
28.605093  
28.808512  
28.60507  
28.808495  
28.605055  
28.808472  
28.605034  
28.80845  
28.605011  
28.808432  
28.60499  
28.80841  
28.604967  
28.808388  
28.604946  
28.80837  
28.604925  
28.808344  
28.604902  
28.808323  
28.604881  
28.808302  
28.604862  
28.80828  
28.604841  
28.808258  
28.604818  
28.808237  
28.6048  
28.808216  
28.604778  
28.808195  
28.604757  
28.808174  
28.604734  
28.808153  
28.604713  
28.80813  
28.60469  
28.808107

28.604673  
28.808088  
28.604649  
28.808067  
28.604631  
28.808046  
28.60461  
28.80803  
28.604586  
28.808008  
28.604567  
28.807981  
28.604546  
28.807966  
28.604523  
28.80794  
28.604502  
28.80792  
28.60448  
28.807896  
28.604462  
28.807878  
28.604443  
28.80786  
28.60442  
28.807838  
28.604397  
28.807814  
28.604372  
28.807793  
28.604355  
28.807774  
28.604334  
28.807753  
28.604313  
28.807732  
28.60429  
28.807709  
28.60427  
28.807688  
28.60425  
28.807667  
28.604229  
28.807646  
28.60421

28.807625  
28.604185  
28.807602  
28.604166  
28.80758  
28.604143  
28.80756  
28.604122  
28.807539  
28.604101  
28.807518  
28.604078  
28.807497  
28.604057  
28.807474  
28.604038  
28.807453  
28.604017  
28.807432  
28.603992  
28.807411  
28.603973  
28.80739  
28.603952  
28.807367  
28.603933  
28.80735  
28.60391  
28.807325  
28.603888  
28.807304  
28.603868  
28.807287  
28.603846  
28.807264  
28.603825  
28.80724  
28.603802  
28.807219  
28.603786  
28.807205  
28.60376  
28.807177  
28.60374  
28.807158

28.603722  
28.807138  
28.603699  
28.807117  
28.603676  
28.80709  
28.603657  
28.807072  
28.603634  
28.807049  
28.603615  
28.807026  
28.603596  
28.807009  
28.603573  
28.806984  
28.603548  
28.806963  
28.603527  
28.806944  
28.60351  
28.80692  
28.603489  
28.806904  
28.603464  
28.806877  
28.603445  
28.806856  
28.603426  
28.806835  
28.603405  
28.806816  
28.60338  
28.806795  
28.603361  
28.806776  
28.603336  
28.80675  
28.603321  
28.806734  
28.603292  
28.806707  
28.603277  
28.806684  
28.603254

28.806671  
28.603233  
28.806648  
28.603212  
28.806627  
28.603193  
28.806608  
28.603168  
28.806583  
28.603148  
28.80656  
28.603128  
28.806541  
28.60311  
28.80652  
28.603085  
28.8065  
28.603065  
28.80648  
28.603045  
28.806456  
28.603022  
28.806435  
28.603  
28.806414  
28.60298  
28.806393  
28.602957  
28.80637  
28.602936  
28.806349  
28.602917  
28.806328  
28.602896  
28.806307  
28.602877  
28.80629  
28.602852  
28.806267  
28.602833  
28.806242  
28.60281  
28.806221  
28.602789  
28.8062

28.60277  
28.806185  
28.602749  
28.80616  
28.602724  
28.806135  
28.602703  
28.806114  
28.602686  
28.806095  
28.602661  
28.806072  
28.60264  
28.806051  
28.602621  
28.806032  
28.6026  
28.806007  
28.602577  
28.805992  
28.602552  
28.805965  
28.602535  
28.805946  
28.602512  
28.805927  
28.602491  
28.805904  
28.602472  
28.80588  
28.60245  
28.805859  
28.602432  
28.805838  
28.60241  
28.805819  
28.602388  
28.805798  
28.60237  
28.805779  
28.602348  
28.805758  
28.602325  
28.805737  
28.602304

28.805714  
28.60228  
28.805693  
28.602262  
28.805672  
28.60224  
28.80565  
28.60222  
28.80563  
28.602198  
28.805607  
28.602177  
28.805586  
28.602152  
28.805565  
28.602133  
28.805544  
28.602114  
28.805523  
28.60209  
28.805498  
28.602068  
28.805477  
28.602047  
28.805452  
28.602024  
28.805431  
28.602003  
28.805416  
28.601988  
28.805395  
28.601965  
28.805372  
28.601944  
28.805351  
28.601925  
28.805336  
28.6019  
28.805311  
28.60188  
28.805292  
28.60186  
28.805271  
28.601835  
28.805248

28.601818  
28.805225  
28.601795  
28.805208  
28.60177  
28.805182  
28.601753  
28.805164  
28.601732  
28.805138  
28.601711  
28.805117  
28.601688  
28.805098  
28.601667  
28.805077  
28.601646  
28.805058  
28.601627  
28.80503  
28.601604  
28.805014  
28.601583  
28.80499  
28.60156  
28.80497  
28.60154  
28.804949  
28.60152  
28.80493  
28.6015  
28.804907  
28.601477  
28.804888  
28.601456  
28.804867  
28.601435  
28.804846  
28.601416  
28.804823  
28.601397  
28.804802  
28.601372  
28.804781  
28.601353

28.804762  
28.601332  
28.80474  
28.601309  
28.804716  
28.601288  
28.804695  
28.601267  
28.804674  
28.601244  
28.804647  
28.601225  
28.804632  
28.601204  
28.804607  
28.601183  
28.804588  
28.601164  
28.804565  
28.601141  
28.804546  
28.60112  
28.804522  
28.601099  
28.804504  
28.601076  
28.804482  
28.601055  
28.80446  
28.601032  
28.80444  
28.601017  
28.804419  
28.600992  
28.804398  
28.600973  
28.80438  
28.600948  
28.804354  
28.600927  
28.804333  
28.600906  
28.804312  
28.600883  
28.80429

28.600864  
28.804272  
28.600843  
28.804247  
28.600822  
28.804226  
28.6008  
28.804205  
28.600779  
28.804184  
28.60076  
28.804165  
28.600735  
28.80414  
28.60072  
28.804123  
28.600695  
28.804098  
28.600676  
28.804077  
28.600657  
28.80406  
28.600632  
28.804033  
28.60061  
28.804018  
28.600594  
28.803997  
28.600573  
28.803976  
28.600548  
28.80395  
28.600523  
28.803926  
28.600504  
28.80391  
28.600483  
28.803886  
28.600458  
28.803865  
28.600443  
28.803846  
28.60042  
28.803825  
28.600405

28.803804  
28.600382  
28.803783  
28.600355  
28.803759  
28.60034  
28.803743  
28.600315  
28.803719  
28.600296  
28.803701  
28.600271  
28.803675  
28.600252  
28.803656  
28.600227  
28.803637  
28.600208  
28.803616  
28.600187  
28.80359  
28.600166  
28.80357  
28.600147  
28.80355  
28.600124  
28.80353  
28.600101  
28.803505  
28.600082  
28.80349  
28.600063  
28.803467  
28.60004  
28.803444  
28.600018  
28.803421  
28.599998  
28.8034  
28.599976  
28.803377  
28.599955  
28.803356  
28.599934  
28.803333

28.599915  
28.803314  
28.599892  
28.803293  
28.599873  
28.803276  
28.59985  
28.80325  
28.59983  
28.803228  
28.599808  
28.803207  
28.599787  
28.803188  
28.599764  
28.803164  
28.599747  
28.803146  
28.599724  
28.803125  
28.599703  
28.803102  
28.59968  
28.80308  
28.59966  
28.803066  
28.599638  
28.80304  
28.599619  
28.803019  
28.599596  
28.803  
28.599575  
28.802979  
28.599554  
28.802956  
28.599531  
28.802935  
28.59951  
28.802914  
28.599493  
28.802893  
28.599472  
28.802872  
28.599451

28.802849  
28.599422  
28.802828  
28.599407  
28.802807  
28.599386  
28.802786  
28.599363  
28.802765  
28.599342  
28.802744  
28.59932  
28.802721  
28.599298  
28.802706  
28.599277  
28.802681  
28.599258  
28.802658  
28.59924  
28.802639  
28.599216  
28.802614  
28.599195  
28.802597  
28.599174  
28.802572  
28.599155  
28.802551  
28.59913  
28.802534  
28.59911  
28.802511  
28.599087  
28.802486  
28.599068  
28.802471  
28.599047  
28.80245  
28.599022  
28.802423  
28.599  
28.802402  
28.598982  
28.802383

28.598963  
28.802359  
28.598942  
28.80234  
28.598919  
28.802322  
28.598898  
28.802296  
28.598875  
28.802273  
28.598854  
28.802252  
28.598835  
28.80223  
28.598816  
28.802212  
28.598795  
28.802193  
28.598772  
28.80217  
28.598751  
28.80215  
28.598732  
28.80213  
28.598707  
28.802103  
28.598686  
28.802084  
28.598665  
28.802061  
28.598648  
28.802044  
28.598621  
28.802017  
28.5986  
28.801996  
28.598583  
28.801977  
28.598562  
28.801958  
28.598537  
28.801931  
28.598518  
28.801916  
28.598497

28.801895  
28.598475  
28.801874  
28.598454  
28.80185  
28.598436  
28.80183  
28.598412  
28.80181  
28.598389  
28.801788  
28.598366  
28.801767  
28.59835  
28.801746  
28.598328  
28.801723  
28.598305  
28.801702  
28.598286  
28.801682  
28.598263  
28.80166  
28.598242  
28.80164  
28.59822  
28.801617  
28.598204  
28.8016  
28.598183  
28.801575  
28.598156  
28.801554  
28.598133  
28.801533  
28.598118  
28.801514  
28.598095  
28.801493  
28.59807  
28.801468  
28.598053  
28.801449  
28.59803  
28.801426

28.598011  
28.801407  
28.597986  
28.801386  
28.597971  
28.801365  
28.597944  
28.80134  
28.597927  
28.80132  
28.5979  
28.801298  
28.597881  
28.80128  
28.597862  
28.801258  
28.597841  
28.801233  
28.597818  
28.801214  
28.5978  
28.801199  
28.597776  
28.801172  
28.597757  
28.801153  
28.597734  
28.801132  
28.597715  
28.801113  
28.597694  
28.80109  
28.597673  
28.801071  
28.597649  
28.801046  
28.59763  
28.801023  
28.59761  
28.801004  
28.597586  
28.800983  
28.597565  
28.80096  
28.597548

28.80094  
28.597523  
28.800919  
28.597506  
28.800894  
28.597483  
28.800877  
28.597464  
28.800854  
28.597437  
28.800827  
28.597418  
28.800806  
28.597397  
28.800787  
28.597378  
28.80077  
28.597353  
28.800745  
28.597332  
28.800726  
28.597313  
28.800705  
28.597288  
28.800684  
28.59727  
28.800663  
28.59725  
28.800642  
28.597229  
28.80062  
28.597206  
28.800598  
28.597185  
28.800577  
28.597164  
28.800556  
28.597145  
28.800535  
28.59712  
28.800512  
28.597101  
28.800491  
28.597078  
28.80047

28.597057  
28.80045  
28.597038  
28.800428  
28.59702  
28.80041  
28.596992  
28.800383  
28.596975  
28.80037  
28.596954  
28.800348  
28.59693  
28.800323  
28.596909  
28.800304  
28.59689  
28.800282  
28.59687  
28.800262  
28.59685  
28.800241  
28.59682  
28.800215  
28.596807  
28.800198  
28.596785  
28.80017  
28.596762  
28.800156  
28.59674  
28.80013  
28.596724  
28.800114  
28.596703  
28.800093  
28.596678  
28.80007  
28.596653  
28.800047  
28.596638  
28.800028  
28.596617  
28.800007  
28.596592

28.799986  
28.596573  
28.799963  
28.59655  
28.799942  
28.59653  
28.799921  
28.596508  
28.7999  
28.596485  
28.79988  
28.596464  
28.799856  
28.596447  
28.799835  
28.596424  
28.799816  
28.5964  
28.799793  
28.596382  
28.799772  
28.596361  
28.799751  
28.59634  
28.799732  
28.596317  
28.799706  
28.596302  
28.799686  
28.596277  
28.799667  
28.596252  
28.799644  
28.596231  
28.799622  
28.596214  
28.799606  
28.596193  
28.79958  
28.59617  
28.79956  
28.59615  
28.799541  
28.596128  
28.799519

28.596106  
28.7995  
28.596085  
28.799473  
28.596066  
28.799458  
28.596045  
28.799435  
28.596025  
28.799412  
28.596003  
28.799387  
28.595976  
28.799366  
28.59596  
28.79935  
28.595942  
28.79933  
28.595915  
28.799303  
28.595896  
28.799284  
28.595873  
28.799265  
28.595858  
28.799244  
28.595829  
28.799223  
28.595812  
28.7992  
28.595793  
28.79918  
28.59577  
28.799158  
28.595749  
28.799137  
28.595724  
28.799112  
28.595705  
28.799093  
28.595684  
28.799072  
28.595661  
28.799051  
28.59564

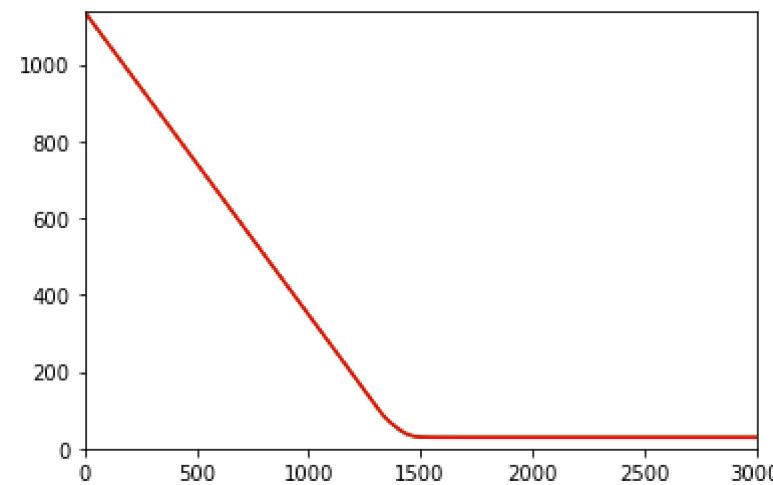
28.79903  
28.59562  
28.79901  
28.5956  
28.79899  
28.595575  
28.798965  
28.595554  
28.798944  
28.595537  
28.798923  
28.595516  
28.798903  
28.595493  
28.798882  
28.595472  
28.798859  
28.595453  
28.798841  
28.595428  
28.798817  
28.595407  
28.798798  
28.595385  
28.798775  
28.59537  
28.798752  
28.595348  
28.79873  
28.595324  
28.79871  
28.595304  
28.798689  
28.595285  
28.798668  
28.595264  
28.798649  
28.59524  
28.798628  
28.595217  
28.798603  
28.595196  
28.798582  
28.595177  
28.798567

28.595152  
28.79854  
28.59513  
28.798517  
28.595112  
28.798502  
28.595093  
28.798481  
28.595072  
28.798456  
28.595049  
28.798433  
28.595028  
28.798414  
28.595009  
28.798395  
28.594984  
28.798372  
28.594965  
28.798353  
28.594944  
28.798332  
28.594925  
28.798307  
28.594904  
28.798288  
28.594881  
28.798267  
28.59486  
28.798246  
28.594837  
28.798225  
28.594816  
28.798203  
28.594795  
28.798182  
28.594772  
28.79816  
28.594753  
28.79814  
28.59473  
28.798113  
28.594707  
28.798096  
28.594692

28.798073  
28.594667  
28.798054  
28.594648  
28.79803  
28.594627  
28.798012  
28.594604  
28.79799  
28.594584  
28.797966  
28.594563  
28.797947  
28.594543  
28.797926  
28.59452  
28.797905  
28.594501  
28.797884  
28.594477  
28.797861  
28.594454  
28.79784  
28.594437  
28.797821  
28.594412  
28.797794  
28.594393  
28.797777  
28.594372  
28.797754  
28.59435  
28.797733  
28.594328  
28.797712  
28.594307  
28.797691  
28.594286  
28.79767  
28.594267  
28.797653  
28.594242  
28.797626  
28.594223  
28.797606

28.594204  
28.797585  
28.594183  
28.797564  
28.59416  
28.797543  
28.59414  
28.797525  
28.594118  
28.797504  
28.594099  
28.797483  
28.59408  
28.797462  
28.594057  
28.79744  
28.594032  
28.797417  
28.594015  
28.797396  
28.593992  
28.797377  
28.593971  
28.797356  
28.593948  
28.797329  
28.593927  
28.79731  
28.593906  
28.797285  
28.593884  
28.797264  
28.593864  
28.797245  
28.593845  
28.797224  
28.593822  
28.797203  
28.5938  
28.797182  
28.593775  
28.797161  
28.59376  
28.797136  
28.593739

```
28.797121  
28.593716  
28.797098  
28.593695  
28.797071  
28.593674  
28.797056  
28.593655  
28.797035  
28.59363  
28.797016  
28.593613  
28.796995  
28.593592  
28.796976
```



In [ ]: