In [3]:
```
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)
```

Task I

In [4]:
```
# Ingest data 2015-2022
from pyspark import SparkFiles

players_15 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_
players_16 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_
players_17 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_
players_18 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_
players_19 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_
players_20 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_
players_21 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_
players_22 = spark.read.csv('gs://dataproc-staging-us-west1-682917987486-wdaku6r5/FIFA_
```

In [5]:
```
print(players_15.columns[55])
```

```
movement_agility
```

In [6]:
```
#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 [7]:
```
#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('_')
players_16 = players_16.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_')
players_17 = players_17.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_')
players_18 = players_18.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_')
players_19 = players_19.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_')
players_20 = players_20.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_')
players_21 = players_21.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_')
players_22 = players_22.withColumn('data_ID', sf.concat(sf.col('sofifa_id'),sf.lit('_')
```

In [8]:

```
# 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 [9]:

```
# db_properties={}
# #update your db username
# db_properties['username']="postgres"
# #update your db password
# db_properties['password']="psql"
# #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 [10]:

```
data.printSchema()
```

```
root
 |-- sofifa_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: string (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: string (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 [11]:

```
# 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)
```

In [13]:

```
data_read = data
```

Task II

Find X

In [33]:

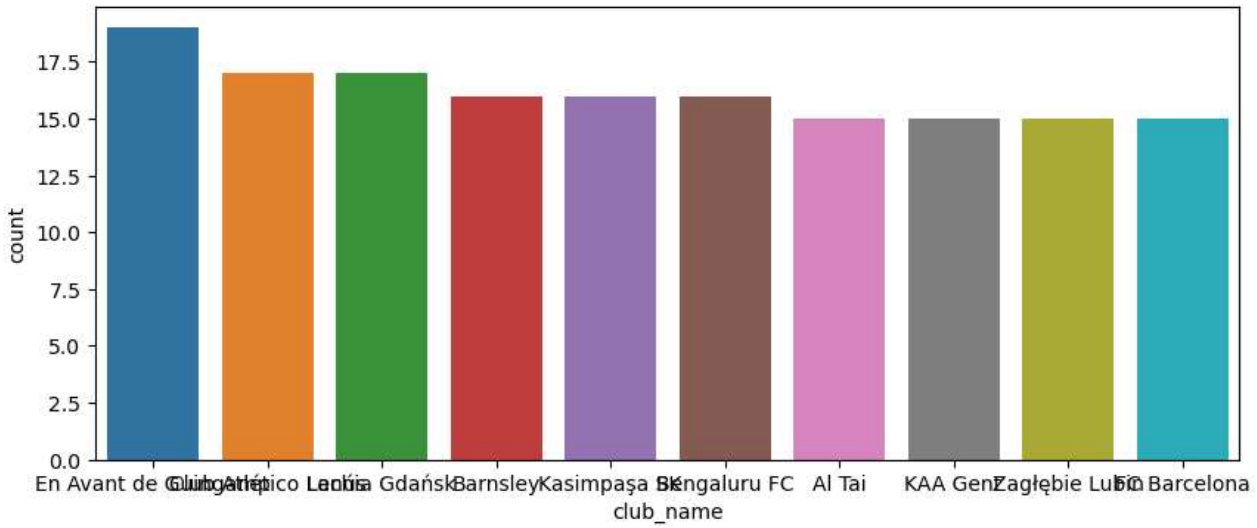
```
#In order to find the club with highest numbers of players
#use where to find the data in 2022
#use .agg to Compute aggregates when(data_read["club_contract_valid_until"]=='2023' and
#group the data by club name and sort them by the count of data
from pyspark.sql import functions as F
def findX(x):
```

```
    SORT_club = data_read.where(data_read['Year']=='2022').groupBy("club_name")
    highest_number_club = SORT_club.agg(F.count(F.when(data_read["club_contract_valid_u
    highest_number_club.show(x,vertical = True)
findX(5)
```

```
-RECORD 0-------------------------
 club_name | En Avant de Guingamp
 count     | 19
-RECORD 1-------------------------
 club_name | Lechia Gdańsk
 count     | 17
-RECORD 2-------------------------
 club_name | Club Atlético Lanús
 count     | 17
-RECORD 3-------------------------
 club_name | Barnsley
 count     | 16
-RECORD 4-------------------------
 club_name | Kasimpaşa SK
 count     | 16
only showing top 5 rows
```

In [34]:

```
# 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

SORT_club = data_read.where(data_read['Year']=='2022').groupBy("club_name")
highest_number_club = SORT_club.agg(F.count(F.when(data_read["club_contract_valid_until
plt.figure( figsize = ( 10, 4 ) )
sns.barplot( x="club_name", y="count", data=highest_number_club[:10])
plt.show()
```



FindY

In [35]:

```
# the number of players older than 27 years oldfor each club
# we used where to find the players who are older than 27 and group the data by club_na
# 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")
highest_number_over27_sum = data_read.where(data_read['age']>27).groupBy("club_name").s
print(highest_number_over27_count)
print(highest_number_over27_sum)
average_over27 = highest_number_over27_sum['sum(age)'].values/highest_number_over27_cou
print(type(average_over27),average_over27)
```

```
[Stage 38:================================================>       (25 + 4) / 29]
                  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              Santiago Wanderers       749
2              1. FC Union Berlin      2215
3                           Carpi       554
4     Club Independiente Santa Fe      2119
...                           ...       ...
1007                    Gefle IF       614
1008                        Bury      1731
1009           Accrington Stanley      1020
1010                 Bohemian FC      1269
1011                        Como       219

[1012 rows x 2 columns]
<class 'numpy.ndarray'> [1.26430206e+00 5.63157895e+00 1.87711864e+01 ... 5.10000000e+02
 1.26900000e+03 2.19000000e+02]
```

In [36]:

```
# we calculated the average and found the highest average
def findY(y):
    highest_number_over27_avg = data_read.where(data_read['age']>27).groupBy("club_name
    highest_number_over27_avg.show(y,vertical =True)
findY(5)
```

```
[Stage 41:=============================================>          (23 + 4) / 29]
-RECORD 0------------------------
 club_name | Yokohama FC
 avg(age)  | 34.7037037037037
-RECORD 1------------------------
 club_name | Wexford Youths
 avg(age)  | 34.0
-RECORD 2------------------------
 club_name | Zamora Fútbol Club
 avg(age)  | 33.857142857142854
-RECORD 3------------------------
 club_name | Centro Atlético F...
 avg(age)  | 33.6
-RECORD 4------------------------
 club_name | CF Fuenlabrada
 avg(age)  | 33.54545454545455
```

```
only showing top 5 rows
```

In [37]:
```
top1_highest_number_over27_avg = data_read.where(data_read['age']>27).groupBy("club_nam
print("club with highest numbers over 27:",top1_highest_number_over27_avg['club_name'].
findY(1)
```

```
club with highest numbers over 27: ['Yokohama FC']
[Stage 48:=============================================>           (23 + 4) / 29]
-RECORD 0--------------------
 club_name | Yokohama FC
 avg(age)  | 34.7037037037037
only showing top 1 row
```

In [38]:
```
# most frequent nation_positions for each year
# we also used where to filter the year, and group them by nation_position, count the n
# them to search the highest one.
most_frequent_nation_2015 = data_read.where(data_read['Year']==2015).groupBy("nation_po
print("Most frequent nation position 2015:",most_frequent_nation_2015['nation_position'
most_frequent_nation_2016 = data_read.where(data_read['Year']==2016).groupBy("nation_po
print("Most frequent nation position 2016:",most_frequent_nation_2016['nation_position'
most_frequent_nation_2017 = data_read.where(data_read['Year']==2017).groupBy("nation_po
print("Most frequent nation position 2017:",most_frequent_nation_2017['nation_position'
most_frequent_nation_2018 = data_read.where(data_read['Year']==2018).groupBy("nation_po
print("Most frequent nation position 2018:",most_frequent_nation_2018['nation_position'
most_frequent_nation_2019 = data_read.where(data_read['Year']==2019).groupBy("nation_po
print("Most frequent nation position 2019:",most_frequent_nation_2019['nation_position'
most_frequent_nation_2020 = data_read.where(data_read['Year']==2020).groupBy("nation_po
print("Most frequent nation position 2020:",most_frequent_nation_2020['nation_position'
most_frequent_nation_2021 = data_read.where(data_read['Year']==2021).groupBy("nation_po
print("Most frequent nation position 2021:",most_frequent_nation_2021['nation_position'
most_frequent_nation_2022 = data_read.where(data_read['Year']==2022).groupBy("nation_po
print("Most frequent nation position 2022:",most_frequent_nation_2022['nation_position'
```

```
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 [39]:
```
# most_frequent_nation_2015 = data_read.where(data_read['Year']==2015).groupBy("nation_
# print(most_frequent_nation_2015)
```

Task III

Data Engineering and cleaning

In [40]:

```
# Since we only use the skillsets of the players, we dropped all the unneccessary colum
drop_list = ['player_position',' nationality_name','work_rate','body_type','real_face',
                              'club_loaned_from','club_joined','club_contract_valid_until
                              'nationality_id','nation_team_id','nation_position','nation
                              'release_clause_eur','player_tags','player_traits','player_
                              'club_logo_url','club_flag_url','nation_logo_url','nation_f
                              'player_url','mentality_composure','goalkeeping_speed','clu
```

In [41]:
```
print(len(drop_list))
```

```
33
```

In [42]:
```
df_read = data_read.drop('player_positions','nationality_name','work_rate','body_type',
                              'club_loaned_from','club_joined','club_contract_valid_until
                              'nationality_id','nation_team_id','nation_position','nation
                              'release_clause_eur','player_tags','player_traits','player_
                              'club_logo_url','club_flag_url','nation_logo_url','nation_f
                              'player_url','mentality_composure','goalkeeping_speed','clu
```

In [43]:
```
print(len(data_read.columns))
print(len(df_read.columns))
```

```
112
79
```

In [44]:
```
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 = false)
```

In [45]:

```
# 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)).ali
#                         for c in df_read.columns])

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

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

In [46]:

```
# we checked the missing values of the whole dataset and we found out that the columns
```

```
# larger amoungt of missing value: 'pace','shooting','passing','dribbling','defending',
# Because there were many missing value in the data, we want to impute the missing valu
```

In [47]:

```
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 [48]:

```
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 [49]:

```
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 [50]:

```
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","dribbli
```

In [51]:

```
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","defendi
```

In [52]:

```
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 [53]:

```
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(
#                         for c in df_read.columns])

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

drop null

In [54]:

```
# we chaecked 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 t
```

In [55]:

```
df_read = df_fully_imputed6.dropna()
```

In [56]:

```
null_counts_plays_df = df_read.select([count(when(isnan(c) | col(c).isNull(), c)).alias
                        for c in df_read.columns])

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

```
22/11/29 06:14:00 WARN org.apache.spark.sql.catalyst.util.package: Truncated the string
representation of a plan since it was too large. This behavior can be adjusted by settin
g 'spark.sql.debug.maxToStringFields'.

-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 [57]:
```
# Because there are only two values in the 'preferred_foot' column, we convert the valu
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 [58]:
```
# df_read_bi.show(vertical = True)
```

string to int

In [59]:
```
# We found the format "XX+X" in some columns and we calculated them and converted them
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 [60]:
```
from pyspark.sql.functions import udf
from pyspark.sql.types import IntegerType
```

In [61]:
```
addallUDF = udf(lambda i:addall(i), IntegerType())
```

In [62]:
```
str_to_int_list = ['ls','st','rs','lw','lf','cf','rf','rw','lam','cam','ram','lm','lcm'
                   'ldm','cdm','rdm','rwb','lb','lcb','cb','rcb','rb','gk']
```

In [63]:
```
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 [64]:
```
print(df_read_int.columns)
```

```
['sofifa_id', 'overall', 'potential', 'value_eur', 'wage_eur', 'age', 'height_cm', 'weig
ht_kg', 'preferred_foot', 'weak_foot', 'skill_moves', 'international_reputation', 'attac
king_crossing', 'attacking_finishing', 'attacking_heading_accuracy', 'attacking_short_pa
ssing', 'attacking_volleys', 'skill_dribbling', 'skill_curve', 'skill_fk_accuracy', 'ski
ll_long_passing', 'skill_ball_control', 'movement_acceleration', 'movement_sprint_spee
d', 'movement_agility', 'movement_reactions', 'movement_balance', 'power_shot_power', 'p
ower_jumping', 'power_stamina', 'power_strength', 'power_long_shots', 'mentality_aggress
ion', 'mentality_interceptions', 'mentality_positioning', 'mentality_vision', 'mentality
_penalties', 'defending_marking_awareness', 'defending_standing_tackle', 'defending_slid
ing_tackle', 'goalkeeping_diving', 'goalkeeping_handling', 'goalkeeping_kicking', 'goalk
eeping_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', 'dri
bbling', 'defending', 'physic']
```

In [65]:
```
# we changed the type of data to DoubleType
```

In [66]:
```
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 [67]:
```
df_read_int.printSchema()
```

```
root
 |-- sofifa_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 = false)
|-- 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 [68]:
```
df_read_features = df_read_int.withColumn('overall_new',col('overall')).drop('overall')
```

In [69]:
```
df_read_int.show(5,vertical = True)
```

```
[Stage 110:>                                                        (0 + 1) / 1]
-RECORD 0------------------------------
 sofifa_id                     | 158023.0
 overall                       | 93.0
 potential                     | 95.0
 value_eur                     | 1.005E8
 wage_eur                      | 550000.0
 age                           | 27.0
 height_cm                     | 169.0
 weight_kg                     | 67.0
 preferred_foot                | 0.0
 weak_foot                     | 3.0
 skill_moves                   | 4.0
 international_reputation      | 5.0
 attacking_crossing            | 84.0
 attacking_finishing           | 94.0
 attacking_heading_accuracy    | 71.0
 attacking_short_passing       | 89.0
 attacking_volleys             | 85.0
 skill_dribbling               | 96.0
 skill_curve                   | 89.0
 skill_fk_accuracy             | 90.0
 skill_long_passing            | 76.0
 skill_ball_control            | 96.0
 movement_acceleration         | 96.0
 movement_sprint_speed         | 90.0
 movement_agility              | 94.0
 movement_reactions            | 94.0
 movement_balance              | 95.0
 power_shot_power              | 80.0
 power_jumping                 | 73.0
 power_stamina                 | 77.0
 power_strength                | 60.0
 power_long_shots              | 88.0
 mentality_aggression          | 48.0
 mentality_interceptions       | 22.0
 mentality_positioning         | 92.0
 mentality_vision              | 90.0
 mentality_penalties           | 76.0
 defending_marking_awareness   | 25.0
 defending_standing_tackle     | 21.0
 defending_sliding_tackle      | 20.0
 goalkeeping_diving            | 6.0
 goalkeeping_handling          | 11.0
 goalkeeping_kicking           | 15.0
 goalkeeping_positioning       | 14.0
 goalkeeping_reflexes          | 8.0
 ls                            | 92.0
 st                            | 92.0
 rs                            | 92.0
 lw                            | 95.0
 lf                            | 93.0
 cf                            | 93.0
 rf                            | 93.0
 rw                            | 95.0
```

```
lam                          | 95.0
cam                          | 95.0
ram                          | 95.0
lm                           | 93.0
lcm                          | 82.0
cm                           | 82.0
rcm                          | 82.0
rm                           | 93.0
lwb                          | 65.0
ldm                          | 65.0
cdm                          | 65.0
rdm                          | 65.0
rwb                          | 65.0
lb                           | 57.0
lcb                          | 48.0
cb                           | 48.0
rcb                          | 48.0
rb                           | 57.0
gk                           | 18.0
Year                         | 2015.0
pace                         | 93.0
shooting                     | 89.0
passing                      | 86.0
dribbling                    | 96.0
defending                    | 27.0
physic                       | 63.0
-RECORD 1-------------------------------
sofifa_id                    | 20801.0
overall                      | 92.0
potential                    | 92.0
value_eur                    | 7.9E7
wage_eur                     | 375000.0
age                          | 29.0
height_cm                    | 185.0
weight_kg                    | 80.0
preferred_foot               | 1.0
weak_foot                    | 4.0
skill_moves                  | 5.0
international_reputation     | 5.0
attacking_crossing           | 83.0
attacking_finishing          | 95.0
attacking_heading_accuracy   | 86.0
attacking_short_passing      | 82.0
attacking_volleys            | 87.0
skill_dribbling              | 93.0
skill_curve                  | 88.0
skill_fk_accuracy            | 79.0
skill_long_passing           | 72.0
skill_ball_control           | 92.0
movement_acceleration        | 91.0
movement_sprint_speed        | 94.0
movement_agility             | 93.0
movement_reactions           | 90.0
movement_balance             | 63.0
power_shot_power             | 94.0
power_jumping                | 94.0
power_stamina                | 89.0
power_strength               | 79.0
power_long_shots             | 93.0
mentality_aggression         | 63.0
mentality_interceptions      | 24.0
mentality_positioning        | 91.0
mentality_vision             | 81.0
mentality_penalties          | 85.0
defending_marking_awareness  | 22.0
```

```
defending_standing_tackle   | 31.0
defending_sliding_tackle    | 23.0
goalkeeping_diving          | 7.0
goalkeeping_handling        | 11.0
goalkeeping_kicking         | 15.0
goalkeeping_positioning     | 14.0
goalkeeping_reflexes        | 11.0
ls                          | 92.0
st                          | 92.0
rs                          | 92.0
lw                          | 92.0
lf                          | 92.0
cf                          | 92.0
rf                          | 92.0
rw                          | 92.0
lam                         | 92.0
cam                         | 92.0
ram                         | 92.0
lm                          | 90.0
lcm                         | 80.0
cm                          | 80.0
rcm                         | 80.0
rm                          | 90.0
lwb                         | 66.0
ldm                         | 66.0
cdm                         | 66.0
rdm                         | 66.0
rwb                         | 66.0
lb                          | 60.0
lcb                         | 55.0
cb                          | 55.0
rcb                         | 55.0
rb                          | 60.0
gk                          | 19.0
Year                        | 2015.0
pace                        | 93.0
shooting                    | 93.0
passing                     | 81.0
dribbling                   | 91.0
defending                   | 32.0
physic                      | 79.0
-RECORD 2-------------------------------
sofifa_id                   | 9014.0
overall                     | 90.0
potential                   | 90.0
value_eur                   | 5.45E7
wage_eur                    | 275000.0
age                         | 30.0
height_cm                   | 180.0
weight_kg                   | 80.0
preferred_foot              | 0.0
weak_foot                   | 2.0
skill_moves                 | 4.0
international_reputation    | 5.0
attacking_crossing          | 80.0
attacking_finishing         | 85.0
attacking_heading_accuracy  | 50.0
attacking_short_passing     | 86.0
attacking_volleys           | 86.0
skill_dribbling             | 93.0
skill_curve                 | 85.0
skill_fk_accuracy           | 83.0
skill_long_passing          | 76.0
skill_ball_control          | 90.0
movement_acceleration       | 93.0
```

```
movement_sprint_speed       | 93.0
movement_agility            | 93.0
movement_reactions          | 89.0
movement_balance            | 91.0
power_shot_power            | 86.0
power_jumping               | 61.0
power_stamina               | 78.0
power_strength              | 65.0
power_long_shots            | 90.0
mentality_aggression        | 47.0
mentality_interceptions     | 39.0
mentality_positioning       | 89.0
mentality_vision            | 84.0
mentality_penalties         | 80.0
defending_marking_awareness | 29.0
defending_standing_tackle   | 26.0
defending_sliding_tackle    | 26.0
goalkeeping_diving          | 10.0
goalkeeping_handling        | 8.0
goalkeeping_kicking         | 11.0
goalkeeping_positioning     | 5.0
goalkeeping_reflexes        | 15.0
ls                          | 87.0
st                          | 87.0
rs                          | 87.0
lw                          | 90.0
lf                          | 90.0
cf                          | 90.0
rf                          | 90.0
rw                          | 90.0
lam                         | 90.0
cam                         | 90.0
ram                         | 90.0
lm                          | 90.0
lcm                         | 81.0
cm                          | 81.0
rcm                         | 81.0
rm                          | 90.0
lwb                         | 67.0
ldm                         | 67.0
cdm                         | 67.0
rdm                         | 67.0
rwb                         | 67.0
lb                          | 58.0
lcb                         | 49.0
cb                          | 49.0
rcb                         | 49.0
rb                          | 58.0
gk                          | 17.0
Year                        | 2015.0
pace                        | 93.0
shooting                    | 86.0
passing                     | 83.0
dribbling                   | 92.0
defending                   | 32.0
physic                      | 64.0
-RECORD 3-------------------------------
sofifa_id                   | 41236.0
overall                     | 90.0
potential                   | 90.0
value_eur                   | 5.25E7
wage_eur                    | 275000.0
age                         | 32.0
height_cm                   | 195.0
weight_kg                   | 95.0
```

preferred_foot	1.0
weak_foot	4.0
skill_moves	4.0
international_reputation	5.0
attacking_crossing	76.0
attacking_finishing	91.0
attacking_heading_accuracy	76.0
attacking_short_passing	84.0
attacking_volleys	92.0
skill_dribbling	88.0
skill_curve	80.0
skill_fk_accuracy	80.0
skill_long_passing	76.0
skill_ball_control	90.0
movement_acceleration	74.0
movement_sprint_speed	77.0
movement_agility	86.0
movement_reactions	85.0
movement_balance	41.0
power_shot_power	93.0
power_jumping	72.0
power_stamina	78.0
power_strength	93.0
power_long_shots	88.0
mentality_aggression	84.0
mentality_interceptions	20.0
mentality_positioning	86.0
mentality_vision	83.0
mentality_penalties	91.0
defending_marking_awareness	25.0
defending_standing_tackle	41.0
defending_sliding_tackle	27.0
goalkeeping_diving	13.0
goalkeeping_handling	15.0
goalkeeping_kicking	10.0
goalkeeping_positioning	9.0
goalkeeping_reflexes	12.0
ls	90.0
st	90.0
rs	90.0
lw	87.0
lf	89.0
cf	89.0
rf	89.0
rw	87.0
lam	89.0
cam	89.0
ram	89.0
lm	86.0
lcm	79.0
cm	79.0
rcm	79.0
rm	86.0
lwb	64.0
ldm	68.0
cdm	68.0
rdm	68.0
rwb	64.0
lb	59.0
lcb	58.0
cb	58.0
rcb	58.0
rb	59.0
gk	20.0
Year	2015.0

```
pace                        | 76.0
shooting                    | 91.0
passing                     | 81.0
dribbling                   | 86.0
defending                   | 34.0
physic                      | 86.0
-RECORD 4-------------------------------
sofifa_id                   | 167495.0
overall                     | 90.0
potential                   | 90.0
value_eur                   | 6.35E7
wage_eur                    | 300000.0
age                         | 28.0
height_cm                   | 193.0
weight_kg                   | 92.0
preferred_foot              | 1.0
weak_foot                   | 4.0
skill_moves                 | 1.0
international_reputation    | 5.0
attacking_crossing          | 25.0
attacking_finishing         | 25.0
attacking_heading_accuracy  | 25.0
attacking_short_passing     | 42.0
attacking_volleys           | 25.0
skill_dribbling             | 25.0
skill_curve                 | 25.0
skill_fk_accuracy           | 25.0
skill_long_passing          | 41.0
skill_ball_control          | 31.0
movement_acceleration       | 58.0
movement_sprint_speed       | 61.0
movement_agility            | 43.0
movement_reactions          | 89.0
movement_balance            | 35.0
power_shot_power            | 42.0
power_jumping               | 78.0
power_stamina               | 44.0
power_strength              | 83.0
power_long_shots            | 25.0
mentality_aggression        | 29.0
mentality_interceptions     | 30.0
mentality_positioning       | 25.0
mentality_vision            | 20.0
mentality_penalties         | 37.0
defending_marking_awareness | 25.0
defending_standing_tackle   | 25.0
defending_sliding_tackle    | 25.0
goalkeeping_diving          | 87.0
goalkeeping_handling        | 85.0
goalkeeping_kicking         | 92.0
goalkeeping_positioning     | 90.0
goalkeeping_reflexes        | 86.0
ls                          | 41.0
st                          | 41.0
rs                          | 41.0
lw                          | 39.0
lf                          | 40.0
cf                          | 40.0
rf                          | 40.0
rw                          | 39.0
lam                         | 39.0
cam                         | 39.0
ram                         | 39.0
lm                          | 41.0
lcm                         | 39.0
```

```
 cm                        | 39.0
 rcm                       | 39.0
 rm                        | 41.0
 lwb                       | 39.0
 ldm                       | 43.0
 cdm                       | 43.0
 rdm                       | 43.0
 rwb                       | 39.0
 lb                        | 39.0
 lcb                       | 41.0
 cb                        | 41.0
 rcb                       | 41.0
 rb                        | 39.0
 gk                        | 90.0
 Year                      | 2015.0
 pace                      | 69.0
 shooting                  | 54.0
 passing                   | 58.0
 dribbling                 | 63.0
 defending                 | 55.0
 physic                    | 66.0
only showing top 5 rows

Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/daemon.py", line 186, in manager
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/daemon.py", line 74, in worker
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line 643, in main
    if read_int(infile) == SpecialLengths.END_OF_STREAM:
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 564, in read
_int
    raise EOFError
EOFError
```

In [70]:
```
feature_cols = df_read_int.columns
del feature_cols[2]
```

In [71]:
```
from pyspark.sql import Row
from pyspark.ml.linalg import Vectors
```

In [72]:
```
len(df_read_features.columns)
```

Out[72]: 79

In [73]:
```
null_counts_plays_df_2 = df_read_features.select([count(when(isnan(c) | col(c).isNull()
                        for c in df_read_features.columns])

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

```
[Stage 111:=====================================================> (28 + 1) / 29]
-RECORD 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 [74]:
```
df_read_features = df_read_features.dropna()
```

In [75]:
```
# 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()
                         for c in df_read_features.columns])

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

```
[Stage 114:=====================================================> (28 + 1) / 29]
-RECORD 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 [76]:

```
# we assembled the features in the data to a vector so that it can be recognized by pys
```

In [77]:

```
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()
```

```
[Stage 118:>                                                        (0 + 1) / 1]
+------+--------------------+
|output|            features|
+------+--------------------+
|  93.0|[158023.0,95.0,1....|
|  92.0|[20801.0,92.0,7.9...|
|  90.0|[9014.0,90.0,5.45...|
|  90.0|[41236.0,90.0,5.2...|
|  90.0|[167495.0,90.0,6....|
|  89.0|[41.0,89.0,3.6E7,...|
|  89.0|[176580.0,91.0,4....|
```

```
|  88.0|[7826.0,88.0,4.05...|
|  88.0|[121944.0,88.0,3....|
|  88.0|[156616.0,88.0,3....|
|  88.0|[167397.0,88.0,4....|
|  88.0|[183277.0,90.0,4....|
|  87.0|[121939.0,87.0,2....|
|  87.0|[155862.0,87.0,3....|
|  87.0|[164240.0,87.0,2....|
|  87.0|[168542.0,87.0,3....|
|  87.0|[173731.0,91.0,3....|
|  87.0|[177003.0,87.0,3....|
|  87.0|[188545.0,89.0,4....|
|  86.0|[10535.0,86.0,1.5...|
+------+--------------------+
only showing top 20 rows

Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/daemon.py", line 186, in manager
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/daemon.py", line 74, in worker
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/worker.py", line 643, in main
    if read_int(infile) == SpecialLengths.END_OF_STREAM:
  File "/usr/lib/spark/python/lib/pyspark.zip/pyspark/serializers.py", line 564, in read
_int
    raise EOFError
EOFError
```

In [78]:
```
# we normalized the features by using StandardScaler
```

In [79]:
```
from pyspark.ml.feature import StandardScaler
Scalerizer=StandardScaler().setInputCol("features").setOutputCol("norm_features")
data_norm = Scalerizer.fit(data).transform(data)
```

In [80]:
```
data_norm = data_norm.drop('features')
data_norm = data_norm.withColumn('features',col('norm_features')).drop('norm_features')
```

In [81]:
```
# data_norm.select('features').show(vertical = False)
```

In [82]:
```
# (trainingData, testData) = data_norm.randomSplit([0.8, 0.2])

# then we splited the data in to training and test dataset

(trainingData, testData) = data_norm.randomSplit([0.8, 0.2])
```

In [83]:
```
trainingData.describe().show()
```

```
[Stage 122:================================================>     (26 + 3) / 29]
+-------+-----------------+
|summary|           output|
+-------+-----------------+
|  count|           111655|
|   mean|65.65939725045901|
| stddev|7.060391706034455|
|    min|             40.0|
```

```
|    max|             94.0|
+-------+------------------+
```

In [84]:
```
print("Training Dataset Count: " + str(trainingData.count()))
print("Test Dataset Count: " + str(testData.count()))
```

```
Training Dataset Count: 111655
[Stage 128:=====================================================>   (27 + 2) / 29]
Test Dataset Count: 28075
```

Pyspark

Random Forest

In [ ]:
```
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

rf = RandomForestRegressor(featuresCol = 'features', labelCol = 'output')

# Create ParamGrid for Cross Validation
rf_paramGrid = (ParamGridBuilder()
             .addGrid(rf.numTrees, [50,100,150])
             .addGrid(rf.maxDepth, [3,4,5])
             .build())

evaluator = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")

rf_cv = CrossValidator(estimator=rf, estimatorParamMaps=rf_paramGrid,
                    evaluator=evaluator, numFolds=5)

# Train model.  This also runs the indexer.
model_rf = rf_cv.fit(trainingData)

# Make predictions.
predictions_rf = model_rf.transform(testData)
predictions_rf.show(5)
```

```
22/11/29 06:17:08 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1064.3 KiB
22/11/29 06:17:32 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.5 KiB
22/11/29 06:17:47 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.5 KiB
22/11/29 06:17:52 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1370.1 KiB
22/11/29 06:18:30 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.7 KiB
22/11/29 06:18:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.7 KiB
```

```
22/11/29 06:19:00 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1675.7 KiB
22/11/29 06:20:05 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1064.2 KiB
22/11/29 06:20:29 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.5 KiB
22/11/29 06:20:44 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.5 KiB
22/11/29 06:20:49 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1370.1 KiB
22/11/29 06:21:27 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.7 KiB
22/11/29 06:21:49 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.7 KiB
22/11/29 06:21:57 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1675.6 KiB
22/11/29 06:23:00 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1064.3 KiB
22/11/29 06:23:24 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.6 KiB
22/11/29 06:23:39 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.6 KiB
22/11/29 06:23:43 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1370.3 KiB
22/11/29 06:24:22 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.8 KiB
22/11/29 06:24:45 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.8 KiB
22/11/29 06:24:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1675.7 KiB
22/11/29 06:25:57 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1064.2 KiB
22/11/29 06:26:21 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.5 KiB
22/11/29 06:26:36 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.5 KiB
22/11/29 06:26:41 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1370.2 KiB
22/11/29 06:27:20 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.7 KiB
22/11/29 06:27:43 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.7 KiB
22/11/29 06:27:51 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1675.8 KiB
22/11/29 06:28:55 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1064.2 KiB
22/11/29 06:29:18 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.5 KiB
22/11/29 06:29:33 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1054.5 KiB
22/11/29 06:29:38 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1370.1 KiB
22/11/29 06:30:16 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.7 KiB
22/11/29 06:30:38 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1202.7 KiB
22/11/29 06:30:45 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1675.8 KiB
22/11/29 06:31:56 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1188.9 KiB
22/11/29 06:32:05 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1662.0 KiB
[Stage 775:>                                                        (0 + 1) / 1]
+------+--------------------+-----------------+
|output|            features|       prediction|
```

```
+------+--------------------+-----------------+
|  66.0|[0.09244059313201...|65.04024140795268|
|  66.0|[0.23534812657222...| 63.7360731340364|
|  66.0|[0.91849251614329...|64.46272326892581|
|  66.0|[0.99638146996760...|64.93970138084205|
|  66.0|[1.39127585699721...|62.96560003701531|
+------+--------------------+-----------------+
only showing top 5 rows
```

In [99]:

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

```
[Stage 1210:=============================================>       (25 + 4) / 29]
Root Mean Squared Error (RMSE) on test data = 1.5497
```

Tune the parameters one by one and see the impact of each parameter

In [ ]:

```
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,

# 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)
```

```
22/11/29 06:33:22 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1040.6 KiB
22/11/29 06:33:28 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1356.3 KiB

+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.09244059313201...| 65.06673200137273|
|  66.0|[0.23534812657222...|63.546544155074045|
|  66.0|[0.91849251614329...| 64.71979246753254|
|  66.0|[0.99638146996760...| 65.16721502433619|
|  66.0|[1.39127585699721...| 63.38429258440907|
+------+--------------------+-----------------+
only showing top 5 rows

[Stage 793:=====================================================> (28 + 1) / 29]
```

```
Root Mean Squared Error (RMSE) on test data = 1.5739
```

In [ ]:

```
trainingData.show(5)
```

```
+------+--------------------+
|output|            features|
+------+--------------------+
|  66.0|[0.03820530001504...|
|  66.0|[0.06238073264976...|
|  66.0|[0.28265544798691...|
|  66.0|[0.45799980291182...|
|  66.0|[0.52388800240667...|
+------+--------------------+
only showing top 5 rows
```

In [ ]:

```
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

# 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)
```

```
22/11/29 06:34:40 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1040.6 KiB
22/11/29 06:34:45 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1356.3 KiB
22/11/29 06:34:53 WARN org.apache.spark.scheduler.DAGScheduler: Broadcasting large task
binary with size 1987.1 KiB

+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.09244059313201...|65.25258903230538|
|  66.0|[0.23534812657222...|63.96400077567084|
|  66.0|[0.91849251614329...|65.41231420039252|
|  66.0|[0.99638146996760...| 65.1097204509944|
|  66.0|[1.39127585699721...|64.18709968896937|
+------+--------------------+-----------------+
only showing top 5 rows

[Stage 813:================================================>     (26 + 3) / 29]
Root Mean Squared Error (RMSE) on test data = 1.31114
```

Linear Regression

In [119…

```
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
```

In [125…

```
from pyspark.ml.regression import LinearRegression

lrg2 = LinearRegression(featuresCol = 'features', labelCol = 'output')

# Create ParamGrid for Cross Validation
lrg_paramGrid2 = ParamGridBuilder() \
    .addGrid(lrg.maxIter, [10, 30,200,1000]) \
    .addGrid(lrg.regParam, [0.1, 1, 2]) \
    .build()

evaluator = RegressionEvaluator(
    labelCol="output", predictionCol="prediction", metricName="rmse")

lrg_cv2 = CrossValidator(estimator=lrg2, estimatorParamMaps=lrg_paramGrid2,
                    evaluator=evaluator, numFolds=5)

model_lrg = lrg_cv2.fit(trainingData)

predictions_lrg = model_lrg.transform(testData)
predictions_lrg.show(5)
```

```
22/11/29 11:54:07 WARN org.apache.spark.ml.util.Instrumentation: [ae96b0f4] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:20 WARN org.apache.spark.ml.util.Instrumentation: [ae96b0f4] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:33 WARN org.apache.spark.ml.util.Instrumentation: [3f24f4bd] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:33 WARN org.apache.spark.ml.util.Instrumentation: [3f24f4bd] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:34 WARN org.apache.spark.ml.util.Instrumentation: [1654152a] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:34 WARN org.apache.spark.ml.util.Instrumentation: [1654152a] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:35 WARN org.apache.spark.ml.util.Instrumentation: [718ff6e9] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:36 WARN org.apache.spark.ml.util.Instrumentation: [718ff6e9] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:36 WARN org.apache.spark.ml.util.Instrumentation: [329c4de6] regParam is
```

```
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:37 WARN org.apache.spark.ml.util.Instrumentation: [329c4de6] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:37 WARN org.apache.spark.ml.util.Instrumentation: [e92c7adf] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:38 WARN org.apache.spark.ml.util.Instrumentation: [e92c7adf] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:38 WARN org.apache.spark.ml.util.Instrumentation: [04914b5f] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:39 WARN org.apache.spark.ml.util.Instrumentation: [04914b5f] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:40 WARN org.apache.spark.ml.util.Instrumentation: [d535f33c] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:40 WARN org.apache.spark.ml.util.Instrumentation: [d535f33c] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:41 WARN org.apache.spark.ml.util.Instrumentation: [7ace6f68] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:41 WARN org.apache.spark.ml.util.Instrumentation: [7ace6f68] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:42 WARN org.apache.spark.ml.util.Instrumentation: [9bb09a55] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:42 WARN org.apache.spark.ml.util.Instrumentation: [9bb09a55] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:43 WARN org.apache.spark.ml.util.Instrumentation: [9b8aec3b] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:44 WARN org.apache.spark.ml.util.Instrumentation: [9b8aec3b] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:44 WARN org.apache.spark.ml.util.Instrumentation: [35d892cf] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:45 WARN org.apache.spark.ml.util.Instrumentation: [35d892cf] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:54:45 WARN org.apache.spark.ml.util.Instrumentation: [0b7df616] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:54:58 WARN org.apache.spark.ml.util.Instrumentation: [0b7df616] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:11 WARN org.apache.spark.ml.util.Instrumentation: [aee3c835] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:12 WARN org.apache.spark.ml.util.Instrumentation: [aee3c835] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:12 WARN org.apache.spark.ml.util.Instrumentation: [ccf7be39] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:13 WARN org.apache.spark.ml.util.Instrumentation: [ccf7be39] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:13 WARN org.apache.spark.ml.util.Instrumentation: [c8e4e269] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:14 WARN org.apache.spark.ml.util.Instrumentation: [c8e4e269] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:15 WARN org.apache.spark.ml.util.Instrumentation: [aac627e8] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:15 WARN org.apache.spark.ml.util.Instrumentation: [aac627e8] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:16 WARN org.apache.spark.ml.util.Instrumentation: [c4736313] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:16 WARN org.apache.spark.ml.util.Instrumentation: [c4736313] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:17 WARN org.apache.spark.ml.util.Instrumentation: [b4f565b7] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:17 WARN org.apache.spark.ml.util.Instrumentation: [b4f565b7] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:18 WARN org.apache.spark.ml.util.Instrumentation: [4945eaec] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:18 WARN org.apache.spark.ml.util.Instrumentation: [4945eaec] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:19 WARN org.apache.spark.ml.util.Instrumentation: [7b064382] regParam is
zero, which might cause numerical instability and overfitting.
```

```
22/11/29 11:55:20 WARN org.apache.spark.ml.util.Instrumentation: [7b064382] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:20 WARN org.apache.spark.ml.util.Instrumentation: [953c1c2a] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:21 WARN org.apache.spark.ml.util.Instrumentation: [953c1c2a] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:21 WARN org.apache.spark.ml.util.Instrumentation: [b7a153f7] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:22 WARN org.apache.spark.ml.util.Instrumentation: [b7a153f7] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:23 WARN org.apache.spark.ml.util.Instrumentation: [c7677890] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:23 WARN org.apache.spark.ml.util.Instrumentation: [c7677890] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:24 WARN org.apache.spark.sql.execution.CacheManager: Asked to cache alrea
dy cached data.
22/11/29 11:55:24 WARN org.apache.spark.sql.execution.CacheManager: Asked to cache alrea
dy cached data.
22/11/29 11:55:24 WARN org.apache.spark.ml.util.Instrumentation: [55ff3405] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:24 WARN org.apache.spark.ml.util.Instrumentation: [55ff3405] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:25 WARN org.apache.spark.ml.util.Instrumentation: [67a17cb4] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:25 WARN org.apache.spark.ml.util.Instrumentation: [67a17cb4] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:26 WARN org.apache.spark.ml.util.Instrumentation: [9d191e0c] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:26 WARN org.apache.spark.ml.util.Instrumentation: [9d191e0c] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:27 WARN org.apache.spark.ml.util.Instrumentation: [7b1d8a59] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:27 WARN org.apache.spark.ml.util.Instrumentation: [7b1d8a59] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:28 WARN org.apache.spark.ml.util.Instrumentation: [a1270fd8] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:29 WARN org.apache.spark.ml.util.Instrumentation: [a1270fd8] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:29 WARN org.apache.spark.ml.util.Instrumentation: [62ff0962] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:30 WARN org.apache.spark.ml.util.Instrumentation: [62ff0962] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:30 WARN org.apache.spark.ml.util.Instrumentation: [20f17d7d] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:31 WARN org.apache.spark.ml.util.Instrumentation: [20f17d7d] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:31 WARN org.apache.spark.ml.util.Instrumentation: [811ee0ac] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:32 WARN org.apache.spark.ml.util.Instrumentation: [811ee0ac] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:33 WARN org.apache.spark.ml.util.Instrumentation: [dfa70524] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:33 WARN org.apache.spark.ml.util.Instrumentation: [dfa70524] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:34 WARN org.apache.spark.ml.util.Instrumentation: [686aa0ea] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:34 WARN org.apache.spark.ml.util.Instrumentation: [686aa0ea] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:35 WARN org.apache.spark.ml.util.Instrumentation: [b2435f03] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:35 WARN org.apache.spark.ml.util.Instrumentation: [b2435f03] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:36 WARN org.apache.spark.ml.util.Instrumentation: [d9dea629] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:36 WARN org.apache.spark.ml.util.Instrumentation: [d9dea629] Cholesky sol
```

```
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:55:37 WARN org.apache.spark.ml.util.Instrumentation: [1bb738f9] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:55:50 WARN org.apache.spark.ml.util.Instrumentation: [1bb738f9] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:03 WARN org.apache.spark.ml.util.Instrumentation: [1367c02f] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:04 WARN org.apache.spark.ml.util.Instrumentation: [1367c02f] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:04 WARN org.apache.spark.ml.util.Instrumentation: [bbc3a494] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:05 WARN org.apache.spark.ml.util.Instrumentation: [bbc3a494] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:06 WARN org.apache.spark.ml.util.Instrumentation: [bc6b28f5] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:06 WARN org.apache.spark.ml.util.Instrumentation: [bc6b28f5] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:07 WARN org.apache.spark.ml.util.Instrumentation: [5c1ca465] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:07 WARN org.apache.spark.ml.util.Instrumentation: [5c1ca465] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:08 WARN org.apache.spark.ml.util.Instrumentation: [ed6a9b93] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:08 WARN org.apache.spark.ml.util.Instrumentation: [ed6a9b93] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:09 WARN org.apache.spark.ml.util.Instrumentation: [1ad68398] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:10 WARN org.apache.spark.ml.util.Instrumentation: [1ad68398] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:10 WARN org.apache.spark.ml.util.Instrumentation: [d61e1fce] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:11 WARN org.apache.spark.ml.util.Instrumentation: [d61e1fce] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:12 WARN org.apache.spark.ml.util.Instrumentation: [715d06ee] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:12 WARN org.apache.spark.ml.util.Instrumentation: [715d06ee] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:13 WARN org.apache.spark.ml.util.Instrumentation: [076c7866] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:13 WARN org.apache.spark.ml.util.Instrumentation: [076c7866] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:14 WARN org.apache.spark.ml.util.Instrumentation: [ba934c91] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:14 WARN org.apache.spark.ml.util.Instrumentation: [ba934c91] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:15 WARN org.apache.spark.ml.util.Instrumentation: [833bcfa3] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:16 WARN org.apache.spark.ml.util.Instrumentation: [833bcfa3] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:16 WARN org.apache.spark.sql.execution.CacheManager: Asked to cache alrea
dy cached data.
22/11/29 11:56:16 WARN org.apache.spark.sql.execution.CacheManager: Asked to cache alrea
dy cached data.
22/11/29 11:56:16 WARN org.apache.spark.ml.util.Instrumentation: [d3e08da0] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:17 WARN org.apache.spark.ml.util.Instrumentation: [d3e08da0] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:17 WARN org.apache.spark.ml.util.Instrumentation: [3de2946e] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:18 WARN org.apache.spark.ml.util.Instrumentation: [3de2946e] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:19 WARN org.apache.spark.ml.util.Instrumentation: [03ff76a7] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:19 WARN org.apache.spark.ml.util.Instrumentation: [03ff76a7] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
```

```
22/11/29 11:56:20 WARN org.apache.spark.ml.util.Instrumentation: [8e5f2c07] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:20 WARN org.apache.spark.ml.util.Instrumentation: [8e5f2c07] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:21 WARN org.apache.spark.ml.util.Instrumentation: [1a865988] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:22 WARN org.apache.spark.ml.util.Instrumentation: [1a865988] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:22 WARN org.apache.spark.ml.util.Instrumentation: [e05899ef] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:23 WARN org.apache.spark.ml.util.Instrumentation: [e05899ef] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:24 WARN org.apache.spark.ml.util.Instrumentation: [06335ac7] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:24 WARN org.apache.spark.ml.util.Instrumentation: [06335ac7] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:25 WARN org.apache.spark.ml.util.Instrumentation: [ce80406e] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:25 WARN org.apache.spark.ml.util.Instrumentation: [ce80406e] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:26 WARN org.apache.spark.ml.util.Instrumentation: [7cda66d2] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:26 WARN org.apache.spark.ml.util.Instrumentation: [7cda66d2] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:27 WARN org.apache.spark.ml.util.Instrumentation: [58ddc4ef] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:28 WARN org.apache.spark.ml.util.Instrumentation: [58ddc4ef] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:28 WARN org.apache.spark.ml.util.Instrumentation: [6555c766] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:29 WARN org.apache.spark.ml.util.Instrumentation: [6555c766] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:30 WARN org.apache.spark.ml.util.Instrumentation: [9ad041a2] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:30 WARN org.apache.spark.ml.util.Instrumentation: [9ad041a2] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
22/11/29 11:56:31 WARN org.apache.spark.ml.util.Instrumentation: [5b74f2da] regParam is
zero, which might cause numerical instability and overfitting.
22/11/29 11:56:44 WARN org.apache.spark.ml.util.Instrumentation: [5b74f2da] Cholesky sol
ver failed due to singular covariance matrix. Retrying with Quasi-Newton solver.
[Stage 2443:>                                                       (0 + 1) / 1]
+------+--------------------+-----------------+
|output|            features|       prediction|
+------+--------------------+-----------------+
|  66.0|[0.09244059313201...|69.11041060954167|
|  66.0|[0.23534812657222...|67.63196245212879|
|  66.0|[0.91849251614329...|66.25163843597272|
|  66.0|[0.99638146996760...|67.14784185470705|
|  66.0|[1.39127585699721...|70.64487733468064|
+------+--------------------+-----------------+
only showing top 5 rows
```

In [126…

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

```
[Stage 2444:=============================================>       (25 + 4) / 29]
Root Mean Squared Error (RMSE) on test data = 1.81377
```

Tune one by one to find the impact for each hyperparameter

In [135…

```
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)
```

```
ERROR:py4j.java_gateway:An error occurred while trying to connect to the Java server (12
7.0.0.1:35431)
Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 977, i
n _get_connection
    connection = self.deque.pop()
IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 1115,
in start
    self.socket.connect((self.address, self.port))
ConnectionRefusedError: [Errno 111] Connection refused
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:977, in GatewayC
lient._get_connection(self)
    976 try:
--> 977     connection = self.deque.pop()
    978 except IndexError:

IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:

ConnectionRefusedError                    Traceback (most recent call last)
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1115, in Gatewa
yConnection.start(self)
   1114 try:
-> 1115     self.socket.connect((self.address, self.port))
   1116     self.stream = self.socket.makefile("rb")

ConnectionRefusedError: [Errno 111] Connection refused

During handling of the above exception, another exception occurred:

Py4JNetworkError                          Traceback (most recent call last)
Cell In [135], line 4
      1 from pyspark.ml.regression import LinearRegression
      3 # Define LinearRegression algorithm
----> 4 lrg_1 = LinearRegression(featuresCol = 'features', labelCol = 'output',regParam
=2.0)
      5 # maxIter: int = 100
```

```
      6 model_1 = lrg_1.fit(trainingData)

File /usr/lib/spark/python/pyspark/__init__.py:114, in keyword_only.<locals>.wrapper(self, *args, **kwargs)
    112     raise TypeError("Method %s forces keyword arguments." % func.__name__)
    113 self._input_kwargs = kwargs
--> 114 return func(self, **kwargs)

File /usr/lib/spark/python/pyspark/ml/regression.py:216, in LinearRegression.__init__(self, featuresCol, labelCol, predictionCol, maxIter, regParam, elasticNetParam, tol, fitIntercept, standardization, solver, weightCol, aggregationDepth, loss, epsilon, maxBlockSizeInMB)
    209 """
    210 __init__(self, \\*, featuresCol="features", labelCol="label", predictionCol="prediction", \
    211          maxIter=100, regParam=0.0, elasticNetParam=0.0, tol=1e-6, fitIntercept=True, \
    212          standardization=True, solver="auto", weightCol=None, aggregationDepth=2, \
    213          loss="squaredError", epsilon=1.35, maxBlockSizeInMB=0.0)
    214 """
    215 super(LinearRegression, self).__init__()
--> 216 self._java_obj = self._new_java_obj(
    217     "org.apache.spark.ml.regression.LinearRegression", self.uid)
    218 kwargs = self._input_kwargs
    219 self.setParams(**kwargs)

File /usr/lib/spark/python/pyspark/ml/wrapper.py:64, in JavaWrapper._new_java_obj(java_class, *args)
     62 java_obj = _jvm()
     63 for name in java_class.split("."):
---> 64     java_obj = getattr(java_obj, name)
     65 java_args = [_py2java(sc, arg) for arg in args]
     66 return java_obj(*java_args)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1692, in JVMView.__getattr__(self, name)
   1689 if name == UserHelpAutoCompletion.KEY:
   1690     return UserHelpAutoCompletion()
-> 1692 answer = self._gateway_client.send_command(
   1693     proto.REFLECTION_COMMAND_NAME +
   1694     proto.REFL_GET_UNKNOWN_SUB_COMMAND_NAME + name + "\n" + self._id +
   1695     "\n" + proto.END_COMMAND_PART)
   1696 if answer == proto.SUCCESS_PACKAGE:
   1697     return JavaPackage(name, self._gateway_client, jvm_id=self._id)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1031, in GatewayClient.send_command(self, command, retry, binary)
   1010 def send_command(self, command, retry=True, binary=False):
   1011     """Sends a command to the JVM. This method is not intended to be
   1012        called directly by Py4J users. It is usually called by
   1013        :class:`JavaMember` instances.
   (...)
   1029      if `binary` is `True`.
   1030     """
-> 1031     connection = self._get_connection()
   1032     try:
   1033         response = connection.send_command(command)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:979, in GatewayClient._get_connection(self)
    977     connection = self.deque.pop()
    978 except IndexError:
--> 979     connection = self._create_connection()
    980 return connection
```

```
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:985, in GatewayC
lient._create_connection(self)
    982 def _create_connection(self):
    983     connection = GatewayConnection(
    984         self.gateway_parameters, self.gateway_property)
--> 985     connection.start()
    986     return connection

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1127, in Gatewa
yConnection.start(self)
   1124 msg = "An error occurred while trying to connect to the Java "\
   1125     "server ({0}:{1})".format(self.address, self.port)
   1126 logger.exception(msg)
-> 1127 raise Py4JNetworkError(msg, e)

Py4JNetworkError: An error occurred while trying to connect to the Java server (127.0.0.
1:35431)
```

In [132…

```
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)
```

```
ERROR:py4j.java_gateway:An error occurred while trying to connect to the Java server (12
7.0.0.1:35431)
Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 977, i
n _get_connection
    connection = self.deque.pop()
IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 1115,
in start
    self.socket.connect((self.address, self.port))
ConnectionRefusedError: [Errno 111] Connection refused
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:977, in GatewayC
lient._get_connection(self)
    976 try:
--> 977     connection = self.deque.pop()
    978 except IndexError:

IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:

ConnectionRefusedError                    Traceback (most recent call last)
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1115, in Gatewa
```

```
yConnection.start(self)
   1114 try:
-> 1115     self.socket.connect((self.address, self.port))
   1116     self.stream = self.socket.makefile("rb")

ConnectionRefusedError: [Errno 111] Connection refused

During handling of the above exception, another exception occurred:

Py4JNetworkError                          Traceback (most recent call last)
Cell In [132], line 4
      1 from pyspark.ml.regression import LinearRegression
      3 # Define LinearRegression algorithm
----> 4 lrg_2 = LinearRegression(featuresCol = 'features', labelCol = 'output',maxIter=
10)
      5 model_2 = lrg_2.fit(trainingData)
      6 predictions_2 = model_2.transform(testData)

File /usr/lib/spark/python/pyspark/__init__.py:114, in keyword_only.<locals>.wrapper(sel
f, *args, **kwargs)
    112     raise TypeError("Method %s forces keyword arguments." % func.__name__)
    113 self._input_kwargs = kwargs
--> 114 return func(self, **kwargs)

File /usr/lib/spark/python/pyspark/ml/regression.py:216, in LinearRegression.__init__(se
lf, featuresCol, labelCol, predictionCol, maxIter, regParam, elasticNetParam, tol, fitIn
tercept, standardization, solver, weightCol, aggregationDepth, loss, epsilon, maxBlockSi
zeInMB)
    209 """
    210 __init__(self, \\*, featuresCol="features", labelCol="label", predictionCol="pre
diction", \
    211          maxIter=100, regParam=0.0, elasticNetParam=0.0, tol=1e-6, fitIntercept=
True, \
    212          standardization=True, solver="auto", weightCol=None, aggregationDepth=
2, \
    213          loss="squaredError", epsilon=1.35, maxBlockSizeInMB=0.0)
    214 """
    215 super(LinearRegression, self).__init__()
--> 216 self._java_obj = self._new_java_obj(
    217     "org.apache.spark.ml.regression.LinearRegression", self.uid)
    218 kwargs = self._input_kwargs
    219 self.setParams(**kwargs)

File /usr/lib/spark/python/pyspark/ml/wrapper.py:64, in JavaWrapper._new_java_obj(java_
class, *args)
     62 java_obj = _jvm()
     63 for name in java_class.split("."):
---> 64     java_obj = getattr(java_obj, name)
     65 java_args = [_py2java(sc, arg) for arg in args]
     66 return java_obj(*java_args)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1692, in JVMVie
w.__getattr__(self, name)
   1689 if name == UserHelpAutoCompletion.KEY:
   1690     return UserHelpAutoCompletion()
-> 1692 answer = self._gateway_client.send_command(
   1693     proto.REFLECTION_COMMAND_NAME +
   1694     proto.REFL_GET_UNKNOWN_SUB_COMMAND_NAME + name + "\n" + self._id +
   1695     "\n" + proto.END_COMMAND_PART)
   1696 if answer == proto.SUCCESS_PACKAGE:
   1697     return JavaPackage(name, self._gateway_client, jvm_id=self._id)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1031, in Gatewa
yClient.send_command(self, command, retry, binary)
   1010 def send_command(self, command, retry=True, binary=False):
```

```
   1011     """Sends a command to the JVM. This method is not intended to be
   1012        called directly by Py4J users. It is usually called by
   1013        :class:`JavaMember` instances.
   (...)
   1029      if `binary` is `True`.
   1030     """
-> 1031     connection = self._get_connection()
   1032     try:
   1033         response = connection.send_command(command)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:979, in GatewayC
lient._get_connection(self)
    977     connection = self.deque.pop()
    978 except IndexError:
--> 979     connection = self._create_connection()
    980 return connection

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:985, in GatewayC
lient._create_connection(self)
    982 def _create_connection(self):
    983     connection = GatewayConnection(
    984         self.gateway_parameters, self.gateway_property)
--> 985     connection.start()
    986     return connection

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1127, in Gatewa
yConnection.start(self)
   1124 msg = "An error occurred while trying to connect to the Java "\
   1125     "server ({0}:{1})".format(self.address, self.port)
   1126 logger.exception(msg)
-> 1127 raise Py4JNetworkError(msg, e)

Py4JNetworkError: An error occurred while trying to connect to the Java server (127.0.0.
1:35431)
```

In [133…

```
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)
```

```
ERROR:py4j.java_gateway:An error occurred while trying to connect to the Java server (12
7.0.0.1:35431)
Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 977, i
n _get_connection
    connection = self.deque.pop()
IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 1115,
in start
```

```
    self.socket.connect((self.address, self.port))
ConnectionRefusedError: [Errno 111] Connection refused

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:977, in GatewayC
lient._get_connection(self)
    976 try:
--> 977     connection = self.deque.pop()
    978 except IndexError:

IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:

ConnectionRefusedError                    Traceback (most recent call last)
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1115, in Gatewa
yConnection.start(self)
   1114 try:
-> 1115     self.socket.connect((self.address, self.port))
   1116     self.stream = self.socket.makefile("rb")

ConnectionRefusedError: [Errno 111] Connection refused

During handling of the above exception, another exception occurred:

Py4JNetworkError                          Traceback (most recent call last)
Cell In [133], line 4
      1 from pyspark.ml.regression import LinearRegression
      3 # Define LinearRegression algorithm
----> 4 lrg_3 = LinearRegression(featuresCol = 'features', labelCol = 'output',maxIter=
30)
      5 model_3 = lrg_3.fit(trainingData)
      6 predictions_3 = model_3.transform(testData)

File /usr/lib/spark/python/pyspark/__init__.py:114, in keyword_only.<locals>.wrapper(sel
f, *args, **kwargs)
    112     raise TypeError("Method %s forces keyword arguments." % func.__name__)
    113 self._input_kwargs = kwargs
--> 114 return func(self, **kwargs)

File /usr/lib/spark/python/pyspark/ml/regression.py:216, in LinearRegression.__init__(se
lf, featuresCol, labelCol, predictionCol, maxIter, regParam, elasticNetParam, tol, fitIn
tercept, standardization, solver, weightCol, aggregationDepth, loss, epsilon, maxBlockSi
zeInMB)
    209 """
    210 __init__(self, \\*, featuresCol="features", labelCol="label", predictionCol="pre
diction", \
    211          maxIter=100, regParam=0.0, elasticNetParam=0.0, tol=1e-6, fitIntercept=
True, \
    212          standardization=True, solver="auto", weightCol=None, aggregationDepth=
2, \
    213          loss="squaredError", epsilon=1.35, maxBlockSizeInMB=0.0)
    214 """
    215 super(LinearRegression, self).__init__()
--> 216 self._java_obj = self._new_java_obj(
    217     "org.apache.spark.ml.regression.LinearRegression", self.uid)
    218 kwargs = self._input_kwargs
    219 self.setParams(**kwargs)

File /usr/lib/spark/python/pyspark/ml/wrapper.py:64, in JavaWrapper._new_java_obj(java_
class, *args)
     62 java_obj = _jvm()
     63 for name in java_class.split("."):
---> 64     java_obj = getattr(java_obj, name)
     65 java_args = [_py2java(sc, arg) for arg in args]
```

```
     66 return java_obj(*java_args)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1692, in JVMVie
w.__getattr__(self, name)
   1689 if name == UserHelpAutoCompletion.KEY:
   1690     return UserHelpAutoCompletion()
-> 1692 answer = self._gateway_client.send_command(
   1693     proto.REFLECTION_COMMAND_NAME +
   1694     proto.REFL_GET_UNKNOWN_SUB_COMMAND_NAME + name + "\n" + self._id +
   1695     "\n" + proto.END_COMMAND_PART)
   1696 if answer == proto.SUCCESS_PACKAGE:
   1697     return JavaPackage(name, self._gateway_client, jvm_id=self._id)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1031, in Gatewa
yClient.send_command(self, command, retry, binary)
   1010 def send_command(self, command, retry=True, binary=False):
   1011     """Sends a command to the JVM. This method is not intended to be
   1012        called directly by Py4J users. It is usually called by
   1013        :class:`JavaMember` instances.
   (...)
   1029      if `binary` is `True`.
   1030     """
-> 1031     connection = self._get_connection()
   1032     try:
   1033         response = connection.send_command(command)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:979, in GatewayC
lient._get_connection(self)
    977     connection = self.deque.pop()
    978 except IndexError:
--> 979     connection = self._create_connection()
    980 return connection

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:985, in GatewayC
lient._create_connection(self)
    982 def _create_connection(self):
    983     connection = GatewayConnection(
    984         self.gateway_parameters, self.gateway_property)
--> 985     connection.start()
    986     return connection

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1127, in Gatewa
yConnection.start(self)
   1124 msg = "An error occurred while trying to connect to the Java "\
   1125     "server ({0}:{1})".format(self.address, self.port)
   1126 logger.exception(msg)
-> 1127 raise Py4JNetworkError(msg, e)

Py4JNetworkError: An error occurred while trying to connect to the Java server (127.0.0.
1:35431)
```

In [134…

```
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)
```

```
ERROR:py4j.java_gateway:An error occurred while trying to connect to the Java server (12
7.0.0.1:35431)
Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 977, i
n _get_connection
    connection = self.deque.pop()
IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py", line 1115,
in start
    self.socket.connect((self.address, self.port))
ConnectionRefusedError: [Errno 111] Connection refused
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:977, in GatewayC
lient._get_connection(self)
    976 try:
--> 977     connection = self.deque.pop()
    978 except IndexError:

IndexError: pop from an empty deque

During handling of the above exception, another exception occurred:

ConnectionRefusedError                    Traceback (most recent call last)
File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1115, in Gatewa
yConnection.start(self)
   1114 try:
-> 1115     self.socket.connect((self.address, self.port))
   1116     self.stream = self.socket.makefile("rb")

ConnectionRefusedError: [Errno 111] Connection refused

During handling of the above exception, another exception occurred:

Py4JNetworkError                          Traceback (most recent call last)
Cell In [134], line 4
      1 from pyspark.ml.regression import LinearRegression
      3 # Define LinearRegression algorithm
----> 4 lrg_4 = LinearRegression(featuresCol = 'features', labelCol = 'output',maxIter=
200)
      5 model_4 = lrg_4.fit(trainingData)
      6 predictions_4 = model_4.transform(testData)

File /usr/lib/spark/python/pyspark/__init__.py:114, in keyword_only.<locals>.wrapper(sel
f, *args, **kwargs)
    112     raise TypeError("Method %s forces keyword arguments." % func.__name__)
    113 self._input_kwargs = kwargs
--> 114 return func(self, **kwargs)

File /usr/lib/spark/python/pyspark/ml/regression.py:216, in LinearRegression.__init__(se
lf, featuresCol, labelCol, predictionCol, maxIter, regParam, elasticNetParam, tol, fitIn
tercept, standardization, solver, weightCol, aggregationDepth, loss, epsilon, maxBlockSi
zeInMB)
    209 """
    210 __init__(self, \\*, featuresCol="features", labelCol="label", predictionCol="pre
diction", \
    211          maxIter=100, regParam=0.0, elasticNetParam=0.0, tol=1e-6, fitIntercept=
```

```
True, \
    212         standardization=True, solver="auto", weightCol=None, aggregationDepth=
2, \
    213         loss="squaredError", epsilon=1.35, maxBlockSizeInMB=0.0)
    214 """
    215 super(LinearRegression, self).__init__()
--> 216 self._java_obj = self._new_java_obj(
    217     "org.apache.spark.ml.regression.LinearRegression", self.uid)
    218 kwargs = self._input_kwargs
    219 self.setParams(**kwargs)

File /usr/lib/spark/python/pyspark/ml/wrapper.py:64, in JavaWrapper._new_java_obj(java_
class, *args)
     62 java_obj = _jvm()
     63 for name in java_class.split("."):
---> 64     java_obj = getattr(java_obj, name)
     65 java_args = [_py2java(sc, arg) for arg in args]
     66 return java_obj(*java_args)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1692, in JVMVie
w.__getattr__(self, name)
   1689 if name == UserHelpAutoCompletion.KEY:
   1690     return UserHelpAutoCompletion()
-> 1692 answer = self._gateway_client.send_command(
   1693     proto.REFLECTION_COMMAND_NAME +
   1694     proto.REFL_GET_UNKNOWN_SUB_COMMAND_NAME + name + "\n" + self._id +
   1695     "\n" + proto.END_COMMAND_PART)
   1696 if answer == proto.SUCCESS_PACKAGE:
   1697     return JavaPackage(name, self._gateway_client, jvm_id=self._id)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1031, in Gatewa
yClient.send_command(self, command, retry, binary)
   1010 def send_command(self, command, retry=True, binary=False):
   1011     """Sends a command to the JVM. This method is not intended to be
   1012        called directly by Py4J users. It is usually called by
   1013        :class:`JavaMember` instances.
   (...)
   1029      if `binary` is `True`.
   1030     """
-> 1031     connection = self._get_connection()
   1032     try:
   1033         response = connection.send_command(command)

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:979, in GatewayC
lient._get_connection(self)
    977     connection = self.deque.pop()
    978 except IndexError:
--> 979     connection = self._create_connection()
    980 return connection

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:985, in GatewayC
lient._create_connection(self)
    982 def _create_connection(self):
    983     connection = GatewayConnection(
    984         self.gateway_parameters, self.gateway_property)
--> 985     connection.start()
    986     return connection

File /usr/lib/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py:1127, in Gatewa
yConnection.start(self)
   1124 msg = "An error occurred while trying to connect to the Java "\
   1125     "server ({0}:{1})".format(self.address, self.port)
   1126 logger.exception(msg)
-> 1127 raise Py4JNetworkError(msg, e)
```

```
Py4JNetworkError: An error occurred while trying to connect to the Java server (127.0.0.
1:35431)
```

Tensorflow

In [100…

```
from pyspark.sql.types import *
```

In [101…

```
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 [104…

```
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 [105…

```
print(x_train)
print(y_train)
```

```
tf.Tensor(
[[6.23807311e-02 1.05221472e+01 1.08017780e-01 ... 8.06517029e+00
  1.64716005e+00 5.66285610e+00]
 [2.82655448e-01 1.05221472e+01 2.07394138e-02 ... 6.43171787e+00
  3.48437715e+00 7.18747139e+00]
 [4.57999796e-01 1.05221472e+01 1.72828455e-04 ... 6.22753620e+00
  3.99119544e+00 7.62307549e+00]
 ...
 [7.66767025e+00 1.11598530e+01 2.59242672e-02 ... 5.30871964e+00
  2.47074008e+00 5.22725201e+00]
 [7.66796017e+00 9.88444138e+00 3.11091207e-02 ... 5.92126417e+00
  1.45710313e+00 5.55395508e+00]
 [7.66801834e+00 9.40616226e+00 2.76525524e-02 ... 5.00244713e+00
  1.33039856e+00 5.44505405e+00]], shape=(56194, 78), dtype=float64)
tf.Tensor([66. 66. 66. ... 51. 52. 52.], shape=(56194,), dtype=float64)
```

Neural networks

In [106…

```
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 [107…

```
y_pred = model_nn(x_train)
model_nn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (56194, 78)               6162

 dense_1 (Dense)             (56194, 10)               790

 dense_2 (Dense)             (56194, 10)               110

 dense_3 (Dense)             (56194, 10)               110

 dense_4 (Dense)             (56194, 1)                11

=================================================================
Total params: 7,183
Trainable params: 7,183
Non-trainable params: 0
_________________________________________________________________
```

In [108…

```
print(y_pred)
print(y_train)
```

```
tf.Tensor(
[[-1.3078046 ]
 [-2.1032403 ]
 [-0.68787897]
 ...
 [-1.8970299 ]
 [-2.3687515 ]
 [-2.0279553 ]], shape=(56194, 1), dtype=float32)
tf.Tensor([66. 66. 66. ... 51. 52. 52.], shape=(56194,), dtype=float64)
```

In [110…

```
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),verbo

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

```
Epoch 1/20
1757/1757 - 6s - loss: 1.6969 - mean_squared_error: 1.6961 - val_loss: 1.4290 - val_mean
_squared_error: 1.4325 - 6s/epoch - 3ms/step
Epoch 2/20
1757/1757 - 4s - loss: 1.6463 - mean_squared_error: 1.6463 - val_loss: 1.8893 - val_mean
_squared_error: 1.8931 - 4s/epoch - 3ms/step
Epoch 3/20
1757/1757 - 4s - loss: 1.6520 - mean_squared_error: 1.6520 - val_loss: 1.5368 - val_mean
_squared_error: 1.5380 - 4s/epoch - 3ms/step
Epoch 4/20
```

```
1757/1757 - 5s - loss: 1.5949 - mean_squared_error: 1.5946 - val_loss: 1.3962 - val_mean
_squared_error: 1.3969 - 5s/epoch - 3ms/step
Epoch 5/20
1757/1757 - 4s - loss: 1.5436 - mean_squared_error: 1.5428 - val_loss: 2.2757 - val_mean
_squared_error: 2.2804 - 4s/epoch - 3ms/step
Epoch 6/20
1757/1757 - 4s - loss: 1.4928 - mean_squared_error: 1.4973 - val_loss: 2.5179 - val_mean
_squared_error: 2.5171 - 4s/epoch - 3ms/step
Epoch 7/20
1757/1757 - 4s - loss: 1.4376 - mean_squared_error: 1.4377 - val_loss: 4.1597 - val_mean
_squared_error: 4.1581 - 4s/epoch - 3ms/step
Epoch 8/20
1757/1757 - 4s - loss: 1.3820 - mean_squared_error: 1.3818 - val_loss: 2.2020 - val_mean
_squared_error: 2.2015 - 4s/epoch - 2ms/step
Epoch 9/20
1757/1757 - 4s - loss: 1.3582 - mean_squared_error: 1.3575 - val_loss: 1.0783 - val_mean
_squared_error: 1.0799 - 4s/epoch - 2ms/step
Epoch 10/20
1757/1757 - 4s - loss: 1.3500 - mean_squared_error: 1.3497 - val_loss: 1.3291 - val_mean
_squared_error: 1.3298 - 4s/epoch - 3ms/step
Epoch 11/20
1757/1757 - 4s - loss: 1.2423 - mean_squared_error: 1.2431 - val_loss: 3.1723 - val_mean
_squared_error: 3.1713 - 4s/epoch - 3ms/step
Epoch 12/20
1757/1757 - 4s - loss: 1.2329 - mean_squared_error: 1.2326 - val_loss: 2.0219 - val_mean
_squared_error: 2.0276 - 4s/epoch - 2ms/step
Epoch 13/20
1757/1757 - 4s - loss: 1.1417 - mean_squared_error: 1.1417 - val_loss: 2.9505 - val_mean
_squared_error: 2.9495 - 4s/epoch - 2ms/step
Epoch 14/20
1757/1757 - 4s - loss: 1.1635 - mean_squared_error: 1.1639 - val_loss: 1.0200 - val_mean
_squared_error: 1.0214 - 4s/epoch - 2ms/step
Epoch 15/20
1757/1757 - 4s - loss: 1.1085 - mean_squared_error: 1.1081 - val_loss: 1.2490 - val_mean
_squared_error: 1.2507 - 4s/epoch - 2ms/step
Epoch 16/20
1757/1757 - 4s - loss: 1.0944 - mean_squared_error: 1.0942 - val_loss: 1.1478 - val_mean
_squared_error: 1.1491 - 4s/epoch - 2ms/step
Epoch 17/20
1757/1757 - 4s - loss: 1.0823 - mean_squared_error: 1.0819 - val_loss: 0.9469 - val_mean
_squared_error: 0.9475 - 4s/epoch - 2ms/step
Epoch 18/20
1757/1757 - 4s - loss: 1.0775 - mean_squared_error: 1.0772 - val_loss: 1.7955 - val_mean
_squared_error: 1.7984 - 4s/epoch - 2ms/step
Epoch 19/20
1757/1757 - 4s - loss: 1.0429 - mean_squared_error: 1.0426 - val_loss: 1.9373 - val_mean
_squared_error: 1.9425 - 4s/epoch - 2ms/step
Epoch 20/20
1757/1757 - 4s - loss: 1.0872 - mean_squared_error: 1.0877 - val_loss: 0.9661 - val_mean
_squared_error: 0.9673 - 4s/epoch - 3ms/step
4547.805
```

In [111…

```
model_nn.evaluate(x_test,y_test, verbose = 2)
```

```
878/878 - 1s - loss: 0.9305 - mean_squared_error: 0.9319 - 970ms/epoch - 1ms/step
```

Out[111… `[0.9305020570755005, 0.931940495967865]`

In [112…

```
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 = (
                  callbacks=[tf.keras.callbacks.TensorBoard(log_dir=logdir, histogram_f
        accuracy = np.mean(history.history["MSE"])
        model.summary()
    return accuracy
```

In [113…

```
from tensorboard.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 [114…

```
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
        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
1230/1230 - 3s - loss: 27.5363 - MSE: 27.5224 - val_loss: 10.1622 - val_MSE: 10.1631 - 3
s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 8.5006 - MSE: 8.4959 - val_loss: 7.3856 - val_MSE: 7.3862 - 2s/ep
och - 2ms/step
```

```
Epoch 3/5
1230/1230 - 2s - loss: 6.5537 - MSE: 6.5549 - val_loss: 5.5198 - val_MSE: 5.5202 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 5.3720 - MSE: 5.3752 - val_loss: 4.5673 - val_MSE: 4.5676 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 4.6772 - MSE: 4.6787 - val_loss: 5.0986 - val_MSE: 5.0985 - 2s/ep
och - 2ms/step
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 10)                790

 dense_6 (Dense)             (None, 1)                 11

=================================================================
Total params: 801
Trainable params: 801
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 3s - loss: 110.4980 - MSE: 110.4329 - val_loss: 17.8756 - val_MSE: 17.8762 -
3s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 12.1077 - MSE: 12.1199 - val_loss: 8.5591 - val_MSE: 8.5604 - 2s/
epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 7.8804 - MSE: 7.8823 - val_loss: 7.2856 - val_MSE: 7.2865 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 7.0528 - MSE: 7.0514 - val_loss: 6.3664 - val_MSE: 6.3672 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 6.3207 - MSE: 6.3176 - val_loss: 5.6708 - val_MSE: 5.6717 - 2s/ep
och - 2ms/step
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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
1230/1230 - 4s - loss: 32.7926 - MSE: 32.7749 - val_loss: 7.0933 - val_MSE: 7.0923 - 4s/
epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 6.5409 - MSE: 6.5398 - val_loss: 6.6526 - val_MSE: 6.6515 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 5.7437 - MSE: 5.7446 - val_loss: 5.3633 - val_MSE: 5.3625 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 5.0309 - MSE: 5.0302 - val_loss: 5.7998 - val_MSE: 5.7992 - 2s/ep
```

```
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 4.6131 - MSE: 4.6149 - val_loss: 4.2171 - val_MSE: 4.2165 - 2s/ep
och - 2ms/step
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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
1230/1230 - 3s - loss: 403.5477 - MSE: 403.3085 - val_loss: 38.5559 - val_MSE: 38.5548 -
3s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 25.1463 - MSE: 25.1331 - val_loss: 17.4940 - val_MSE: 17.4935 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 13.7679 - MSE: 13.7667 - val_loss: 12.2865 - val_MSE: 12.2861 - 2
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 9.9609 - MSE: 9.9558 - val_loss: 9.2169 - val_MSE: 9.2165 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 8.1566 - MSE: 8.1550 - val_loss: 7.6003 - val_MSE: 7.5998 - 2s/ep
och - 2ms/step
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_11 (Dense)            (None, 10)                790

 dense_12 (Dense)            (None, 1)                 11

=================================================================
Total params: 801
Trainable params: 801
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 3s - loss: 52.7300 - MSE: 52.6986 - val_loss: 10.3107 - val_MSE: 10.3113 - 3
s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 8.6681 - MSE: 8.6640 - val_loss: 7.4981 - val_MSE: 7.4988 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 7.0971 - MSE: 7.0951 - val_loss: 6.4978 - val_MSE: 6.4981 - 2s/ep
och - 2ms/step
Epoch 4/5
```

```
1230/1230 - 2s - loss: 6.0469 - MSE: 6.0444 - val_loss: 5.7743 - val_MSE: 5.7746 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 5.1471 - MSE: 5.1458 - val_loss: 4.7953 - val_MSE: 4.7959 - 2s/ep
och - 2ms/step
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_11 (Dense)            (None, 10)                790

 dense_12 (Dense)            (None, 1)                 11

 dense_13 (Dense)            (None, 10)                20

 dense_14 (Dense)            (None, 1)                 11

=================================================================
Total params: 832
Trainable params: 832
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 40.3829 - MSE: 40.3594 - val_loss: 5.6454 - val_MSE: 5.6460 - 4s/
epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 4.9623 - MSE: 4.9602 - val_loss: 4.8772 - val_MSE: 4.8778 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 4.4289 - MSE: 4.4284 - val_loss: 4.0982 - val_MSE: 4.0986 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 4.0473 - MSE: 4.0466 - val_loss: 4.1046 - val_MSE: 4.1050 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 3.7665 - MSE: 3.7661 - val_loss: 3.6047 - val_MSE: 3.6049 - 2s/ep
och - 2ms/step
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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

=================================================================
Total params: 863
Trainable params: 863
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 143.8477 - MSE: 143.7587 - val_loss: 4.9488 - val_MSE: 4.9489 - 4
s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 4.4289 - MSE: 4.4269 - val_loss: 4.1446 - val_MSE: 4.1444 - 2s/ep
och - 2ms/step
Epoch 3/5
```

```
1230/1230 - 2s - loss: 3.8955 - MSE: 3.8963 - val_loss: 3.5726 - val_MSE: 3.5723 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 3.6309 - MSE: 3.6303 - val_loss: 3.3257 - val_MSE: 3.3256 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 3.4136 - MSE: 3.4129 - val_loss: 4.4667 - val_MSE: 4.4665 - 2s/ep
och - 2ms/step
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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
1230/1230 - 3s - loss: 30.4516 - MSE: 30.4347 - val_loss: 12.4071 - val_MSE: 12.4064 - 3
s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 10.5812 - MSE: 10.5853 - val_loss: 8.9311 - val_MSE: 8.9305 - 2s/
epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 8.5596 - MSE: 8.5563 - val_loss: 9.2008 - val_MSE: 9.2015 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 7.3962 - MSE: 7.4024 - val_loss: 8.4267 - val_MSE: 8.4257 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 6.5788 - MSE: 6.5812 - val_loss: 6.2370 - val_MSE: 6.2364 - 2s/ep
och - 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
1230/1230 - 3s - loss: 1059.6027 - MSE: 1058.9749 - val_loss: 54.2037 - val_MSE: 54.2054
```

```
- 3s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 49.7873 - MSE: 49.7825 - val_loss: 45.6439 - val_MSE: 45.6456 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 41.5399 - MSE: 41.5486 - val_loss: 37.2489 - val_MSE: 37.2505 - 2
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 32.6633 - MSE: 32.6567 - val_loss: 26.9428 - val_MSE: 26.9442 - 2
s/epoch - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 21.1371 - MSE: 21.1334 - val_loss: 15.1663 - val_MSE: 15.1665 - 2
s/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

=================================================================
Total params: 832
Trainable params: 832
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 457.4865 - MSE: 457.2085 - val_loss: 24.5646 - val_MSE: 24.5660 -
4s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 16.6311 - MSE: 16.6263 - val_loss: 11.0462 - val_MSE: 11.0464 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 9.8203 - MSE: 9.8177 - val_loss: 8.5023 - val_MSE: 8.5026 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 8.0256 - MSE: 8.0260 - val_loss: 7.3165 - val_MSE: 7.3168 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 6.9216 - MSE: 6.9203 - val_loss: 6.5625 - val_MSE: 6.5622 - 2s/ep
och - 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

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

```
Epoch 1/5
1230/1230 - 4s - loss: 426.6310 - MSE: 426.3643 - val_loss: 6.6005 - val_MSE: 6.5995 - 4
s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 6.2008 - MSE: 6.1996 - val_loss: 6.1520 - val_MSE: 6.1513 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 5.7126 - MSE: 5.7112 - val_loss: 5.4181 - val_MSE: 5.4176 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 5.2663 - MSE: 5.2667 - val_loss: 5.0948 - val_MSE: 5.0944 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 4.8600 - MSE: 4.8589 - val_loss: 4.5223 - val_MSE: 4.5218 - 2s/ep
och - 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
1230/1230 - 4s - loss: 839.0534 - MSE: 838.5250 - val_loss: 4.6685 - val_MSE: 4.6687 - 4
s/epoch - 4ms/step
Epoch 2/5
1230/1230 - 3s - loss: 4.6721 - MSE: 4.6725 - val_loss: 4.5150 - val_MSE: 4.5153 - 3s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 3s - loss: 4.4820 - MSE: 4.4840 - val_loss: 4.2303 - val_MSE: 4.2304 - 3s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 3s - loss: 4.3314 - MSE: 4.3290 - val_loss: 5.2856 - val_MSE: 5.2855 - 3s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 3s - loss: 4.2046 - MSE: 4.2048 - val_loss: 4.0692 - val_MSE: 4.0692 - 3s/ep
och - 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

 dense_27 (Dense)            (None, 10)                20

 dense_28 (Dense)            (None, 1)                 11

=================================================================
Total params: 925
Trainable params: 925
Non-trainable params: 0
_________________________________________________________________
--- Starting trial: run-WIDTH20-DEPTH3
{'NN_width': 20, 'NN_depth': 3}
Epoch 1/5
1230/1230 - 3s - loss: 644.2581 - MSE: 643.8654 - val_loss: 25.2803 - val_MSE: 25.2775 -
3s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 20.8063 - MSE: 20.7960 - val_loss: 16.5602 - val_MSE: 16.5587 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 13.3278 - MSE: 13.3281 - val_loss: 11.1926 - val_MSE: 11.1921 - 2
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 10.2049 - MSE: 10.2064 - val_loss: 10.6150 - val_MSE: 10.6147 - 2
s/epoch - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 8.7590 - MSE: 8.7564 - val_loss: 7.8629 - val_MSE: 7.8626 - 2s/ep
och - 2ms/step
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_29 (Dense)            (None, 20)                1580

 dense_30 (Dense)            (None, 1)                 21

=================================================================
Total params: 1,601
Trainable params: 1,601
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 3s - loss: 205.3304 - MSE: 205.2190 - val_loss: 35.0859 - val_MSE: 35.0827 -
3s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 20.6651 - MSE: 20.6582 - val_loss: 11.2234 - val_MSE: 11.2220 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 10.3665 - MSE: 10.3707 - val_loss: 9.4567 - val_MSE: 9.4554 - 2s/
epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 8.9934 - MSE: 8.9931 - val_loss: 8.5887 - val_MSE: 8.5881 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 7.6704 - MSE: 7.6758 - val_loss: 6.8214 - val_MSE: 6.8206 - 2s/ep
och - 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

=================================================================
Total params: 1,662
Trainable params: 1,662
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 21.5383 - MSE: 21.5262 - val_loss: 7.1054 - val_MSE: 7.1065 - 4s/
epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 7.0655 - MSE: 7.0633 - val_loss: 7.5264 - val_MSE: 7.5275 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 6.4025 - MSE: 6.4040 - val_loss: 7.0139 - val_MSE: 7.0146 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 5.8621 - MSE: 5.8600 - val_loss: 5.2268 - val_MSE: 5.2275 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 5.3507 - MSE: 5.3503 - val_loss: 5.0162 - val_MSE: 5.0169 - 2s/ep
och - 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
1230/1230 - 3s - loss: 717.5007 - MSE: 717.0616 - val_loss: 19.7234 - val_MSE: 19.7215 -
3s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 16.0023 - MSE: 15.9970 - val_loss: 13.3775 - val_MSE: 13.3764 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 11.6068 - MSE: 11.6154 - val_loss: 10.6554 - val_MSE: 10.6545 - 2
s/epoch - 2ms/step
Epoch 4/5
```

```
1230/1230 - 2s - loss: 9.8098 - MSE: 9.8076 - val_loss: 10.2646 - val_MSE: 10.2648 - 2s/
epoch - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 8.6767 - MSE: 8.6784 - val_loss: 8.1003 - val_MSE: 8.1001 - 2s/ep
och - 2ms/step
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_35 (Dense)            (None, 20)                1580

 dense_36 (Dense)            (None, 1)                 21

=================================================================
Total params: 1,601
Trainable params: 1,601
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 3s - loss: 26.1606 - MSE: 26.1498 - val_loss: 8.8660 - val_MSE: 8.8675 - 3s/
epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 8.0701 - MSE: 8.0684 - val_loss: 7.7788 - val_MSE: 7.7804 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 6.7551 - MSE: 6.7526 - val_loss: 6.2460 - val_MSE: 6.2482 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 5.7753 - MSE: 5.7770 - val_loss: 5.2117 - val_MSE: 5.2137 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 4.8395 - MSE: 4.8413 - val_loss: 4.2593 - val_MSE: 4.2609 - 2s/ep
och - 2ms/step
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_35 (Dense)            (None, 20)                1580

 dense_36 (Dense)            (None, 1)                 21

 dense_37 (Dense)            (None, 20)                40

 dense_38 (Dense)            (None, 1)                 21

=================================================================
Total params: 1,662
Trainable params: 1,662
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 6.3936 - MSE: 6.3924 - val_loss: 4.1517 - val_MSE: 4.1513 - 4s/ep
och - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 4.0897 - MSE: 4.0882 - val_loss: 3.7186 - val_MSE: 3.7181 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 3.9035 - MSE: 3.9029 - val_loss: 3.4625 - val_MSE: 3.4621 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 3.6561 - MSE: 3.6589 - val_loss: 3.7643 - val_MSE: 3.7638 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 3.5068 - MSE: 3.5073 - val_loss: 3.1743 - val_MSE: 3.1739 - 2s/ep
och - 2ms/step
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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

=================================================================
Total params: 1,723
Trainable params: 1,723
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 5.5433 - MSE: 5.5423 - val_loss: 3.3183 - val_MSE: 3.3185 - 4s/ep
och - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 3.4029 - MSE: 3.4033 - val_loss: 3.2433 - val_MSE: 3.2434 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 3.3677 - MSE: 3.3673 - val_loss: 3.0802 - val_MSE: 3.0803 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 3s - loss: 3.2777 - MSE: 3.2784 - val_loss: 2.8628 - val_MSE: 2.8630 - 3s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 3.1991 - MSE: 3.1981 - val_loss: 2.8016 - val_MSE: 2.8019 - 2s/ep
och - 2ms/step
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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
1230/1230 - 3s - loss: 305.7193 - MSE: 305.5336 - val_loss: 14.9410 - val_MSE: 14.9445 -
```

```
3s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 12.4871 - MSE: 12.4852 - val_loss: 11.2185 - val_MSE: 11.2211 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 10.1936 - MSE: 10.1903 - val_loss: 9.4809 - val_MSE: 9.4834 - 2s/
epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 8.7776 - MSE: 8.7740 - val_loss: 9.5288 - val_MSE: 9.5310 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 7.5693 - MSE: 7.5680 - val_loss: 6.8149 - val_MSE: 6.8167 - 2s/ep
och - 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
1230/1230 - 3s - loss: 397.5793 - MSE: 397.3423 - val_loss: 35.7416 - val_MSE: 35.7391 -
3s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 24.2156 - MSE: 24.2100 - val_loss: 17.6551 - val_MSE: 17.6558 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 13.2130 - MSE: 13.2071 - val_loss: 12.0210 - val_MSE: 12.0218 - 2
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 10.1065 - MSE: 10.1107 - val_loss: 9.4646 - val_MSE: 9.4655 - 2s/
epoch - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 7.9367 - MSE: 7.9421 - val_loss: 6.5900 - val_MSE: 6.5906 - 2s/ep
och - 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

=================================================================
Total params: 1,662
Trainable params: 1,662
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 164.2424 - MSE: 164.1433 - val_loss: 9.6107 - val_MSE: 9.6107 - 4
s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 8.2497 - MSE: 8.2505 - val_loss: 7.1277 - val_MSE: 7.1275 - 2s/ep
och - 2ms/step
Epoch 3/5
```

```
1230/1230 - 2s - loss: 7.0309 - MSE: 7.0299 - val_loss: 7.3175 - val_MSE: 7.3172 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 6.0536 - MSE: 6.0526 - val_loss: 5.0928 - val_MSE: 5.0927 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 5.2075 - MSE: 5.2049 - val_loss: 4.3759 - val_MSE: 4.3758 - 2s/ep
och - 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

=================================================================
Total params: 1,723
Trainable params: 1,723
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 575.6024 - MSE: 575.2567 - val_loss: 46.6009 - val_MSE: 46.5973 -
4s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 37.2452 - MSE: 37.2328 - val_loss: 25.1273 - val_MSE: 25.1256 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 18.0672 - MSE: 18.0654 - val_loss: 13.6806 - val_MSE: 13.6812 - 2
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 12.5254 - MSE: 12.5225 - val_loss: 12.1792 - val_MSE: 12.1804 - 2
s/epoch - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 9.9247 - MSE: 9.9239 - val_loss: 8.4520 - val_MSE: 8.4529 - 2s/ep
och - 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
1230/1230 - 4s - loss: 741.7123 - MSE: 741.2446 - val_loss: 9.5885 - val_MSE: 9.5885 - 4
s/epoch - 4ms/step
Epoch 2/5
1230/1230 - 3s - loss: 8.4226 - MSE: 8.4283 - val_loss: 9.1449 - val_MSE: 9.1450 - 3s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 3s - loss: 7.6350 - MSE: 7.6360 - val_loss: 7.0578 - val_MSE: 7.0582 - 3s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 3s - loss: 6.7978 - MSE: 6.7969 - val_loss: 6.0846 - val_MSE: 6.0855 - 3s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 3s - loss: 5.9809 - MSE: 5.9821 - val_loss: 6.1859 - val_MSE: 6.1860 - 3s/ep
och - 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

 dense_51 (Dense)            (None, 20)                40

 dense_52 (Dense)            (None, 1)                 21

=================================================================
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
1230/1230 - 3s - loss: 27.1980 - MSE: 27.1877 - val_loss: 11.3945 - val_MSE: 11.3953 - 3
s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 8.9502 - MSE: 8.9511 - val_loss: 9.2056 - val_MSE: 9.2048 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 7.0411 - MSE: 7.0427 - val_loss: 7.0978 - val_MSE: 7.0980 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 5.9376 - MSE: 5.9366 - val_loss: 5.0592 - val_MSE: 5.0586 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 5.3300 - MSE: 5.3296 - val_loss: 5.2413 - val_MSE: 5.2406 - 2s/ep
och - 2ms/step
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_53 (Dense)            (None, 30)                2370

 dense_54 (Dense)            (None, 1)                 31

=================================================================
Total params: 2,401
Trainable params: 2,401
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 31.3136 - MSE: 31.3002 - val_loss: 6.9564 - val_MSE: 6.9556 - 4s/
epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 6.4826 - MSE: 6.4852 - val_loss: 7.4719 - val_MSE: 7.4714 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 5.6057 - MSE: 5.6080 - val_loss: 5.2941 - val_MSE: 5.2935 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 5.1059 - MSE: 5.1056 - val_loss: 5.2356 - val_MSE: 5.2352 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 4.7299 - MSE: 4.7307 - val_loss: 4.1585 - val_MSE: 4.1580 - 2s/ep
och - 2ms/step
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_53 (Dense)            (None, 30)                2370

 dense_54 (Dense)            (None, 1)                 31

 dense_55 (Dense)            (None, 30)                60

 dense_56 (Dense)            (None, 1)                 31

=================================================================
Total params: 2,492
Trainable params: 2,492
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 229.3527 - MSE: 229.2248 - val_loss: 35.4937 - val_MSE: 35.4973 -
4s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 26.4487 - MSE: 26.4362 - val_loss: 21.1871 - val_MSE: 21.1907 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 15.3382 - MSE: 15.3406 - val_loss: 12.8007 - val_MSE: 12.8023 - 2
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 12.1164 - MSE: 12.1143 - val_loss: 11.3335 - val_MSE: 11.3343 - 2
s/epoch - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 9.8173 - MSE: 9.8197 - val_loss: 8.4830 - val_MSE: 8.4836 - 2s/ep
och - 2ms/step
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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

=================================================================
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
1230/1230 - 3s - loss: 100.6722 - MSE: 100.6149 - val_loss: 17.4545 - val_MSE: 17.4529 -
3s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 13.6322 - MSE: 13.6271 - val_loss: 11.0005 - val_MSE: 11.0001 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 9.6612 - MSE: 9.6561 - val_loss: 9.4774 - val_MSE: 9.4774 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 7.8166 - MSE: 7.8170 - val_loss: 9.3424 - val_MSE: 9.3421 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 6.3764 - MSE: 6.3757 - val_loss: 5.3646 - val_MSE: 5.3645 - 2s/ep
och - 2ms/step
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_59 (Dense)            (None, 30)                2370

 dense_60 (Dense)            (None, 1)                 31

=================================================================
Total params: 2,401
Trainable params: 2,401
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 3s - loss: 73.9634 - MSE: 73.9199 - val_loss: 8.0130 - val_MSE: 8.0125 - 3s/
epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 6.5753 - MSE: 6.5747 - val_loss: 5.9241 - val_MSE: 5.9239 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 5.2285 - MSE: 5.2277 - val_loss: 4.6799 - val_MSE: 4.6794 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 4.3727 - MSE: 4.3751 - val_loss: 4.1015 - val_MSE: 4.1010 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 3.9036 - MSE: 3.9026 - val_loss: 3.7064 - val_MSE: 3.7062 - 2s/ep
och - 2ms/step
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 dense_59 (Dense)            (None, 30)                2370

 dense_60 (Dense)            (None, 1)                 31

 dense_61 (Dense)            (None, 30)                60

 dense_62 (Dense)            (None, 1)                 31

=================================================================
Total params: 2,492
Trainable params: 2,492
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 69.9265 - MSE: 69.8850 - val_loss: 8.3880 - val_MSE: 8.3881 - 4s/
epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 6.8763 - MSE: 6.8776 - val_loss: 5.8745 - val_MSE: 5.8741 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 5.6854 - MSE: 5.6835 - val_loss: 5.2165 - val_MSE: 5.2160 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 5.0387 - MSE: 5.0380 - val_loss: 4.6810 - val_MSE: 4.6810 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 4.6998 - MSE: 4.7039 - val_loss: 4.1841 - val_MSE: 4.1839 - 2s/ep
och - 2ms/step
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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

=================================================================
Total params: 2,583
Trainable params: 2,583
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 200.3524 - MSE: 200.2280 - val_loss: 4.2014 - val_MSE: 4.2022 - 4
s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 4.2256 - MSE: 4.2246 - val_loss: 3.6896 - val_MSE: 3.6901 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 3s - loss: 3.9260 - MSE: 3.9267 - val_loss: 4.0227 - val_MSE: 4.0230 - 3s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 3.8445 - MSE: 3.8429 - val_loss: 4.8189 - val_MSE: 4.8191 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 3.6829 - MSE: 3.6826 - val_loss: 3.1282 - val_MSE: 3.1284 - 2s/ep
och - 2ms/step
Model: "sequential_8"
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 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
1230/1230 - 3s - loss: 1189.3732 - MSE: 1188.6248 - val_loss: 18.8719 - val_MSE: 18.8726
- 3s/epoch - 2ms/step
Epoch 2/5
1230/1230 - 2s - loss: 15.8420 - MSE: 15.8451 - val_loss: 13.4293 - val_MSE: 13.4302 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 11.7802 - MSE: 11.7810 - val_loss: 10.7474 - val_MSE: 10.7489 - 2
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 9.9969 - MSE: 9.9942 - val_loss: 9.2234 - val_MSE: 9.2246 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 8.7088 - MSE: 8.7080 - val_loss: 8.0971 - val_MSE: 8.0981 - 2s/ep
och - 2ms/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
1230/1230 - 3s - loss: 14.7645 - MSE: 14.7584 - val_loss: 9.1442 - val_MSE: 9.1448 - 3s/
epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 7.2408 - MSE: 7.2434 - val_loss: 6.6456 - val_MSE: 6.6456 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 5.6680 - MSE: 5.6683 - val_loss: 5.8395 - val_MSE: 5.8399 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 4.6656 - MSE: 4.6638 - val_loss: 4.3125 - val_MSE: 4.3126 - 2s/ep
```

```
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 4.0859 - MSE: 4.0877 - val_loss: 3.6541 - val_MSE: 3.6542 - 2s/ep
och - 2ms/step
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_67 (Dense)            (None, 30)                2370

 dense_68 (Dense)            (None, 1)                 31

 dense_69 (Dense)            (None, 30)                60

 dense_70 (Dense)            (None, 1)                 31

=================================================================
Total params: 2,492
Trainable params: 2,492
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 186.5122 - MSE: 186.3997 - val_loss: 8.0163 - val_MSE: 8.0171 - 4
s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 6.5935 - MSE: 6.5910 - val_loss: 5.3848 - val_MSE: 5.3846 - 2s/ep
och - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 5.2439 - MSE: 5.2424 - val_loss: 4.8342 - val_MSE: 4.8342 - 2s/ep
och - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 4.5819 - MSE: 4.5817 - val_loss: 3.9100 - val_MSE: 3.9099 - 2s/ep
och - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 4.1279 - MSE: 4.1268 - val_loss: 4.8574 - val_MSE: 4.8570 - 2s/ep
och - 2ms/step
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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

=================================================================
Total params: 2,583
Trainable params: 2,583
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 289.8218 - MSE: 289.6737 - val_loss: 46.1233 - val_MSE: 46.1221 -
4s/epoch - 3ms/step
Epoch 2/5
1230/1230 - 2s - loss: 38.9493 - MSE: 38.9523 - val_loss: 29.5118 - val_MSE: 29.5099 - 2
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 2s - loss: 23.6450 - MSE: 23.6408 - val_loss: 16.8819 - val_MSE: 16.8804 - 2
```

```
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 2s - loss: 15.5076 - MSE: 15.4991 - val_loss: 13.7689 - val_MSE: 13.7677 - 2
s/epoch - 2ms/step
Epoch 5/5
1230/1230 - 2s - loss: 12.8180 - MSE: 12.8122 - val_loss: 11.3904 - val_MSE: 11.3894 - 2
s/epoch - 2ms/step
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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

=================================================================
Total params: 2,674
Trainable params: 2,674
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
1230/1230 - 4s - loss: 1042.2133 - MSE: 1041.5867 - val_loss: 56.2574 - val_MSE: 56.2618
- 4s/epoch - 4ms/step
Epoch 2/5
1230/1230 - 3s - loss: 47.4016 - MSE: 47.3957 - val_loss: 41.7383 - val_MSE: 41.7424 - 3
s/epoch - 2ms/step
Epoch 3/5
1230/1230 - 3s - loss: 34.3585 - MSE: 34.3424 - val_loss: 33.2297 - val_MSE: 33.2329 - 3
s/epoch - 2ms/step
Epoch 4/5
1230/1230 - 3s - loss: 20.5470 - MSE: 20.5457 - val_loss: 10.9391 - val_MSE: 10.9387 - 3
s/epoch - 2ms/step
Epoch 5/5
1230/1230 - 3s - loss: 9.0988 - MSE: 9.1018 - val_loss: 7.7555 - val_MSE: 7.7550 - 3s/ep
och - 2ms/step
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 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

=================================================================
Total params: 2,765
Trainable params: 2,765
Non-trainable params: 0
_________________________________________________________________
```

Linear Regression

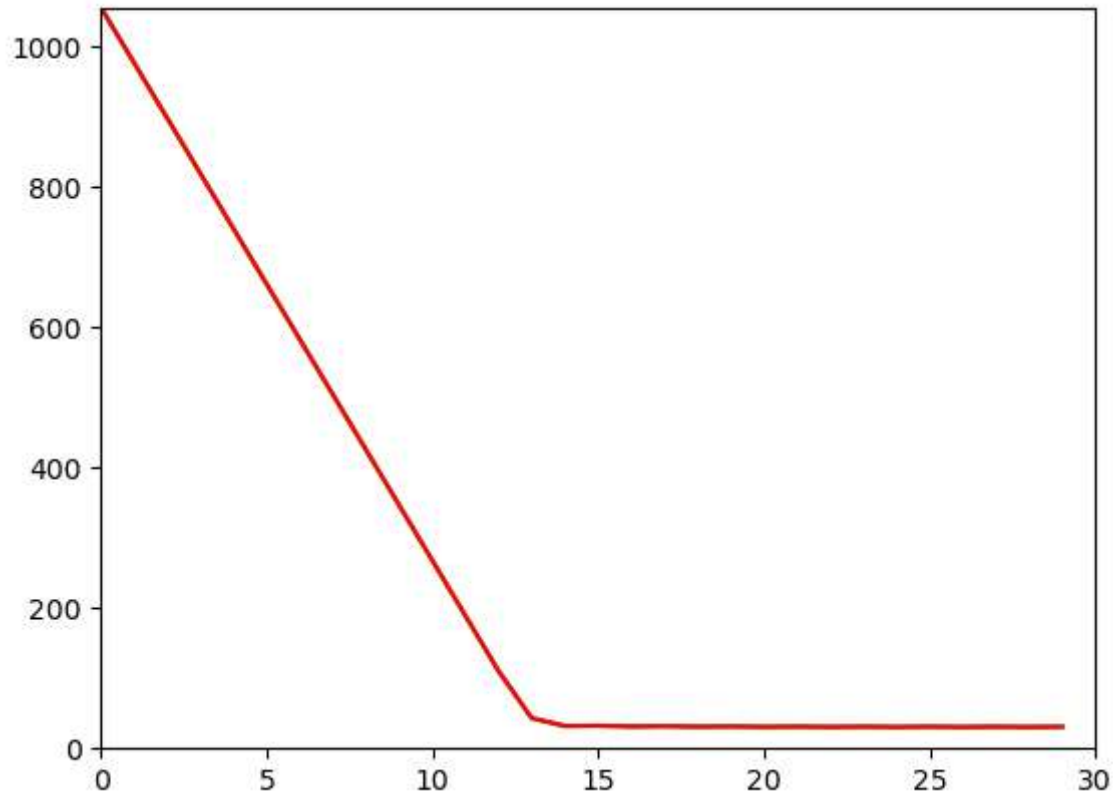
In [115…

```
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
    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

    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 /opt/conda/miniconda3/lib/python3.8/site-packages/tensorflow/pyt
hon/util/tf_should_use.py:243: initialize_all_variables (from tensorflow.python.ops.vari
ables) is deprecated and will be removed after 2017-03-02.
Instructions for updating:
Use `tf.global_variables_initializer` instead.
2022-11-29 11:34:55.200035: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:3
57] MLIR V1 optimization pass is not enabled
1056.6455
1057.1296
977.7627
978.2464
898.8799
899.363
819.9971
820.4797
741.11426
741.5963
662.2315
662.713
583.34863
583.8297
504.4659
504.94635
425.58313
426.06305
346.70035
347.17975
267.81757
268.29642
188.93477
189.41312
110.05222
110.53048
43.02768
43.156796
32.090935
31.787062
32.337605
32.33992
31.340061
31.056177
31.619144
```

```
31.607231
30.997797
30.721834
31.266632
31.245726
30.745256
30.47669
31.014843
30.987577
30.649391
30.384268
30.936522
30.907791
30.612192
30.3485
30.90662
30.877464
30.584766
30.322075
30.868538
30.838861
30.559853
30.298079
30.843607
30.813633
```



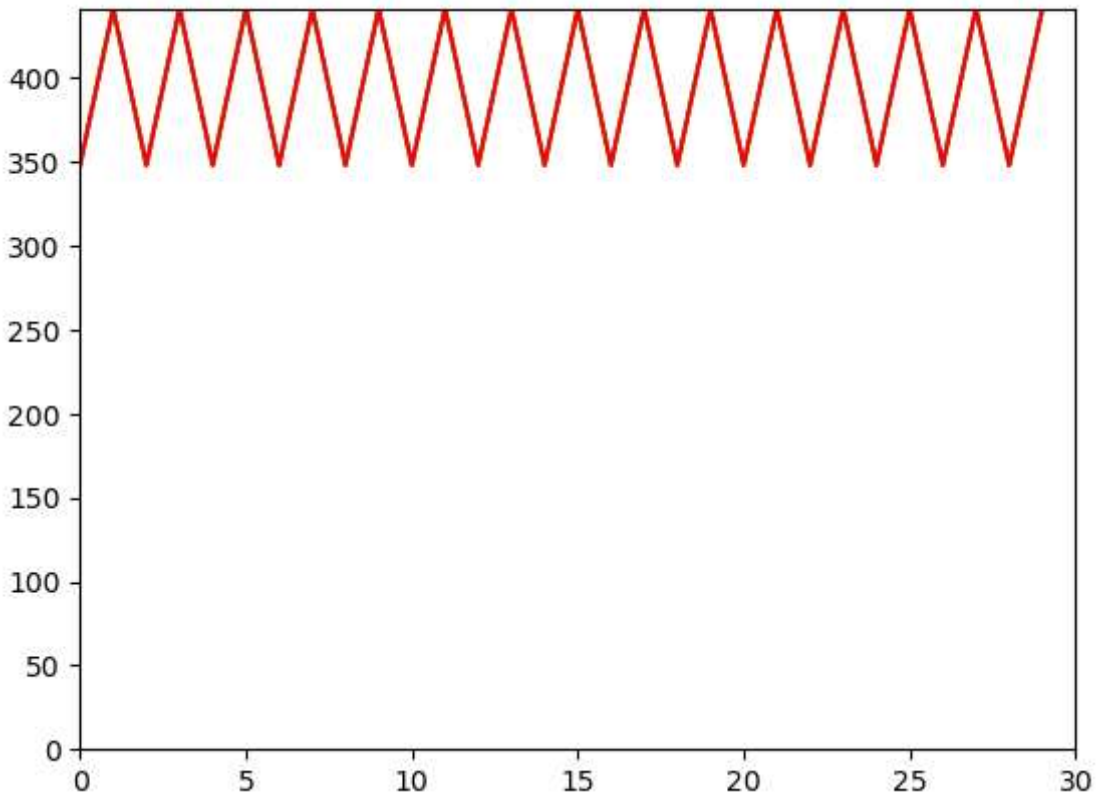
In [116…

```
# 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
    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

    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()
```

```
347.64478
347.93274
441.17865
440.90033
```

```
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
347.64478
347.93274
441.17865
440.90033
```



In [117…

```
# 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
    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

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()
```

```
1129.6991
1129.1133
1121.8109
1121.2249
1113.9229
1113.3369
1106.0348
1105.4487
1098.1465
1097.5605
1090.2583
1089.6724
1082.3702
1081.7842
1074.482
1073.896
1066.5939
1066.0078
1058.7058
1058.1195
1050.8175
1050.2314
1042.9294
1042.3433
1035.0413
1034.4551
1027.1532
1026.5669
1019.265
```

```
1018.6788
1011.37683
1010.7906
1003.4888
1002.9024
995.6005
995.0142
987.71246
987.1261
979.8243
979.2379
971.93616
971.34973
964.04803
963.46155
956.1598
955.5733
948.27167
947.68506
940.3835
939.79694
932.4954
931.9087
924.6072
924.0206
916.71906
916.1324
908.8309
908.2443
900.9428
900.35614
893.05457
892.46783
885.16644
884.5797
877.2784
876.69147
869.39014
868.80347
861.50195
860.9152
853.61383
853.027
845.72577
845.13885
837.8375
837.2506
829.94934
829.3626
822.0612
821.4744
814.1731
813.58606
806.285
805.6979
798.3968
797.8097
790.5087
789.92175
782.6205
782.0334
774.7324
774.1454
766.8442
766.25714
```

```
758.956
758.36896
751.0679
750.4808
743.1799
742.5925
735.2916
734.70435
727.40344
726.8162
719.5153
718.928
711.62714
711.0399
703.739
703.1517
695.8508
695.2636
687.96277
687.37537
680.0745
679.4871
672.1864
671.59906
664.2982
663.7109
656.4101
655.8227
648.522
647.93445
640.6338
640.0462
632.7457
632.1582
624.8575
624.2701
616.9694
616.3817
609.0812
608.4935
601.19305
600.60535
593.30493
592.7172
585.41675
584.82904
577.5286
576.94086
569.64044
569.0527
561.7523
561.16455
553.86414
553.2764
545.976
545.3882
538.0878
537.50006
530.1997
529.6118
522.3115
521.72363
514.4234
513.83545
506.53525
```

```
505.9474
498.6471
498.05914
490.75894
490.17093
482.87085
482.28278
474.98267
474.39462
467.09454
466.50644
459.20636
458.61826
451.31824
450.73007
443.43008
442.8419
435.5419
434.95374
427.65375
427.06558
419.7656
419.1774
411.87747
411.2892
403.9893
403.40106
396.10117
395.51285
388.21304
387.6247
380.3249
379.73648
372.43674
371.84833
364.54855
363.96017
356.6604
356.072
348.77225
348.1838
340.88412
340.29562
332.99597
332.40747
325.10782
324.5193
317.2197
316.63107
309.33154
308.74292
301.4434
300.8547
293.5552
292.96655
285.66705
285.07843
277.7789
277.19022
269.89078
269.30206
262.00262
261.4139
254.11447
253.52571
```

```
246.22632
245.63754
238.33817
237.74934
230.45003
229.86119
222.56187
221.97302
214.67374
214.08485
206.78558
206.19667
198.89743
198.30847
191.00928
190.4203
183.12112
182.53214
175.23299
174.64397
167.34482
166.7558
159.45668
158.86761
151.56853
150.97942
143.68039
143.09126
135.79224
135.20306
127.90409
127.31492
120.01593
119.42675
112.12786
111.538574
104.25037
103.66182
96.5354
95.9214
89.26398
88.63277
82.45868
81.796585
76.29037
75.60924
70.726494
70.04673
65.5642
64.933784
60.646877
60.0797
55.892956
55.374783
51.299473
50.840225
46.962124
46.564087
43.017464
42.666153
39.546703
39.257275
36.66061
36.423435
34.356552
```

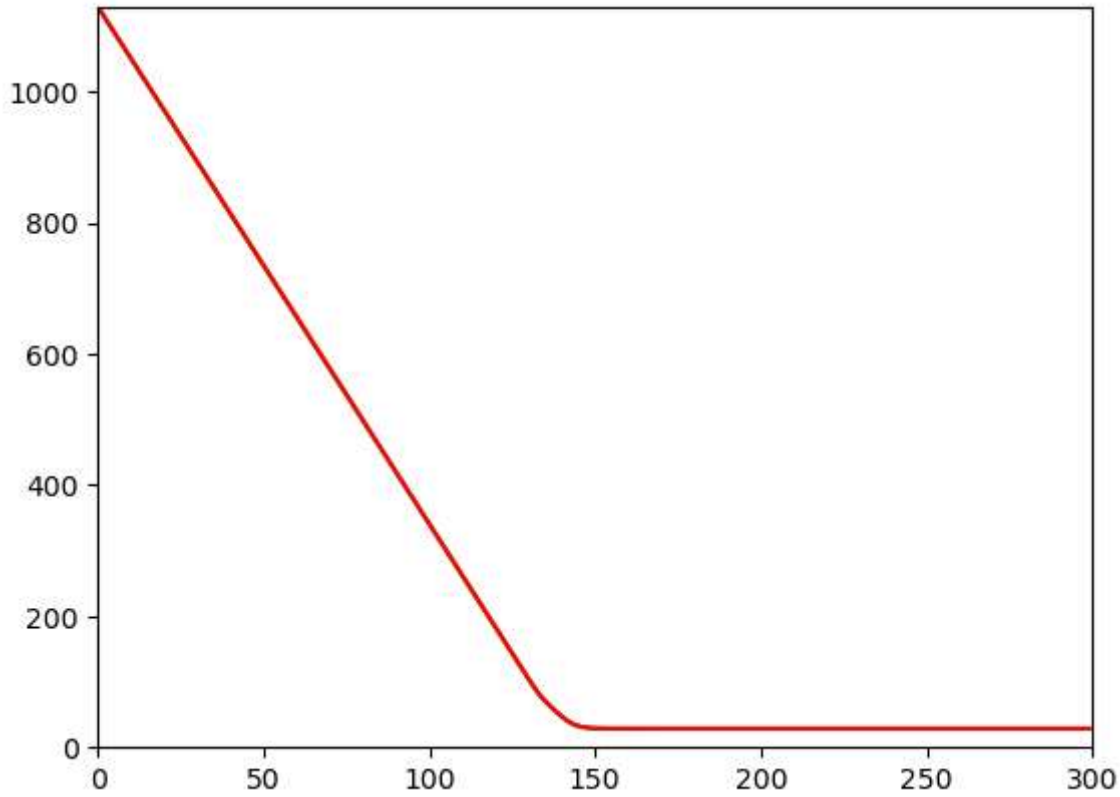
```
34.17705
32.593693
32.477364
31.303913
31.219385
30.391867
30.32659
29.765533
29.722593
29.34201
29.321548
29.061983
29.055267
28.879948
28.882671
28.76458
28.773502
28.690018
28.703533
28.641434
28.657808
28.609121
28.6286
28.58842
28.611225
28.57514
28.600466
28.566956
28.593662
28.561703
28.58948
28.558197
28.58683
28.555922
28.585342
28.554459
28.584414
28.553522
28.583796
28.55279
28.583418
28.55221
28.583141
28.551767
28.582933
28.551413
28.582764
28.551132
28.582605
28.550886
28.582447
28.55065
28.58229
28.550428
28.582119
28.550209
28.581944
28.549995
28.581758
28.54978
28.58157
28.549566
28.581383
28.54935
28.581186
```

```
28.549143
28.58099
28.54893
28.580793
28.548717
28.58059
28.548506
28.580393
28.548296
28.580189
28.548086
28.579987
28.547874
28.579786
28.54766
28.57958
28.547451
28.579374
28.547241
28.57917
28.54703
28.578966
28.546822
28.578758
28.546612
28.578545
28.546398
28.57833
28.54619
28.57812
28.545977
28.577908
28.545767
28.577694
28.545557
28.577484
28.545347
28.577274
28.545135
28.577059
28.544924
28.576849
28.544714
28.57664
28.544504
28.576426
28.544294
28.576218
28.54408
28.576002
28.54387
28.575788
28.543661
28.575577
28.543451
28.575369
28.54324
28.575155
28.543032
28.574945
28.54282
28.574732
28.542606
28.574518
28.542397
```

```
28.574306
28.542187
28.574097
28.541979
28.573887
28.541765
28.573675
28.541553
28.573463
28.541346
28.573252
28.541136
28.573038
28.540926
28.572828
28.540712
28.572617
28.540503
28.572403
28.54029
28.57219
28.540081
28.57198
28.539871
28.571768
28.539661
28.571554
28.53945
28.57134
28.539238
28.571127
28.539028
28.570915
28.538818
28.570705
28.53861
28.570492
28.538395
28.570282
28.538185
28.570066
28.537975
28.569853
28.537766
28.569643
28.537556
28.569433
28.537346
28.569218
28.537134
28.569002
28.536922
28.568794
28.53671
28.568579
28.536503
28.568369
28.536291
28.568155
28.53608
28.567945
28.53587
28.567732
28.53566
28.567522
```

```
28.535448
28.56731
28.535238
28.567091
28.535025
28.566885
28.534817
28.566671
28.534605
28.566462
28.534395
28.566248
28.534185
28.566034
28.533976
28.565825
28.533764
28.565613
28.533554
28.565397
28.533342
28.565186
28.53313
28.564974
28.532923
28.564764
28.53271
28.564548
28.5325
28.564339
28.532291
28.564129
28.53208
28.563917
28.53187
28.563705
28.531656
28.56349
28.531446
28.56328
28.531235
28.563066
28.531025
28.562857
28.530817
28.562641
28.530607
28.56243
28.530394
28.56222
28.530184
28.562008
28.529974
28.561796
28.529764
28.561586
28.529552
28.56137
28.529345
28.561163
28.52913
28.560947
28.52892
28.560734
28.528711
```

```
28.560522
28.5285
28.560308
28.52829
28.5601
28.528076
28.559887
28.527868
28.559677
28.527657
28.559465
28.527449
28.55925
28.527237
28.559038
28.527025
28.558826
28.526815
28.558613
28.526604
28.5584
28.526394
28.55819
28.526184
28.55798
28.525974
28.55776
28.52576
28.55755
28.525549
28.557333
28.525343
28.55713
28.52513
28.556908
28.524921
28.556698
28.524708
28.556488
28.524498
28.556276
28.524288
28.556063
28.524075
28.55585
28.523867
28.555635
28.523653
28.555426
28.523443
28.555208
```



In [118…

```
# 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
    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

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()
```

```
1137.8618
1137.544
1137.073
1136.7551
1136.2843
1135.9663
1135.4954
1135.1776
1134.7067
1134.3888
1133.9177
1133.6
1133.129
1132.8112
1132.3401
1132.0223
1131.5514
1131.2335
1130.7627
1130.4448
1129.9738
1129.656
1129.185
1128.8672
1128.3962
1128.0785
1127.6073
1127.2896
1126.8186
```

```
1126.5009
1126.0298
1125.712
1125.241
1124.9233
1124.4521
1124.1343
1123.6633
1123.3456
1122.8745
1122.5568
1122.0857
1121.7681
1121.2969
1120.9791
1120.5082
1120.1904
1119.7194
1119.4015
1118.9305
1118.6129
1118.1417
1117.824
1117.3528
1117.0353
1116.5641
1116.2465
1115.7754
1115.4576
1114.9865
1114.669
1114.1978
1113.88
1113.4089
1113.0913
1112.6201
1112.3025
1111.8313
1111.5135
1111.0425
1110.7249
1110.2537
1109.936
1109.4648
1109.1473
1108.676
1108.3585
1107.8873
1107.5697
1107.0984
1106.7808
1106.3097
1105.9921
1105.5208
1105.2032
1104.7319
1104.4146
1103.9434
1103.6257
1103.1544
1102.8369
1102.3656
1102.0481
1101.5768
1101.2593
```

```
1100.7881
1100.4705
1099.9991
1099.6816
1099.2104
1098.8928
1098.4216
1098.1041
1097.6328
1097.3153
1096.844
1096.5266
1096.055
1095.7377
1095.2664
1094.9489
1094.4774
1094.1602
1093.6887
1093.3713
1092.8999
1092.5825
1092.1111
1091.7937
1091.3224
1091.0049
1090.5334
1090.2161
1089.7448
1089.4272
1088.9558
1088.6385
1088.1671
1087.8497
1087.3783
1087.0609
1086.5895
1086.2721
1085.8007
1085.4833
1085.0118
1084.6946
1084.223
1083.9058
1083.4342
1083.117
1082.6454
1082.3282
1081.8567
1081.5393
1081.0677
1080.7506
1080.279
1079.9617
1079.4901
1079.173
1078.7014
1078.3842
1077.9126
1077.5953
1077.1238
1076.8068
1076.3351
1076.0177
1075.5463
```

```
1075.229
1074.7574
1074.4402
1073.9685
1073.6514
1073.1798
1072.8627
1072.391
1072.0739
1071.6023
1071.285
1070.8134
1070.4962
1070.0245
1069.7074
1069.2357
1068.9186
1068.4469
1068.1299
1067.6582
1067.3411
1066.8693
1066.5522
1066.0806
1065.7633
1065.2917
1064.9747
1064.5028
1064.1858
1063.7141
1063.3971
1062.9253
1062.6083
1062.1365
1061.8196
1061.3478
1061.0306
1060.559
1060.242
1059.7701
1059.4531
1058.9813
1058.6642
1058.1925
1057.8755
1057.4037
1057.0867
1056.6149
1056.298
1055.826
1055.509
1055.0374
1054.7203
1054.2484
1053.9314
1053.4597
1053.1427
1052.6708
1052.3539
1051.8821
1051.5651
1051.0931
1050.7764
1050.3044
1049.9874
```

```
1049.5156
1049.1987
1048.7268
1048.4098
1047.9381
1047.6211
1047.1493
1046.8324
1046.3605
1046.0436
1045.5717
1045.2548
1044.7828
1044.4658
1043.994
1043.6771
1043.2052
1042.8883
1042.4164
1042.0995
1041.6276
1041.3108
1040.8387
1040.522
1040.05
1039.7332
1039.2611
1038.9443
1038.4724
1038.1555
1037.6835
1037.3666
1036.8948
1036.5781
1036.106
1035.7892
1035.3173
1035.0004
1034.5283
1034.2114
1033.7396
1033.4229
1032.9508
1032.6339
1032.162
1031.8452
1031.3732
1031.0564
1030.5844
1030.2676
1029.7954
1029.4788
1029.0067
1028.69
1028.2179
1027.9012
1027.4291
1027.1124
1026.6403
1026.3236
1025.8516
1025.5348
1025.0626
1024.746
1024.2738
```

```
1023.95715
1023.4851
1023.16846
1022.6963
1022.3796
1021.9075
1021.5909
1021.11865
1020.80194
1020.32983
1020.01324
1019.54095
1019.22437
1018.75214
1018.43555
1017.96344
1017.647
1017.1745
1016.858
1016.3858
1016.06915
1015.597
1015.28046
1014.8082
1014.49164
1014.01935
1013.70294
1013.2305
1012.9141
1012.4418
1012.1252
1011.65295
1011.33655
1010.86414
1010.54767
1010.0753
1009.7588
1009.2865
1008.97015
1008.49774
1008.1813
1007.70905
1007.3926
1006.9201
1006.60376
1006.1314
1005.81494
1005.34247
1005.02606
1004.5538
1004.23737
1003.76483
1003.4485
1002.976
1002.6598
1002.1873
1001.87085
1001.3985
1001.0821
1000.60974
1000.29333
999.82086
999.50446
999.0321
998.71576
```

```
998.2432
997.92694
997.45447
997.1382
996.66565
996.3493
995.8769
995.56055
995.088
994.7718
994.2992
993.983
993.51044
993.19415
992.72156
992.4054
991.9328
991.6166
991.144
990.82776
990.35516
990.039
989.56635
989.2502
988.7775
988.46136
987.98883
987.6726
987.2001
986.8838
986.41125
986.0951
985.62244
985.30615
984.8336
984.51746
984.0447
983.7286
983.256
982.93976
982.46716
982.151
981.6784
981.3622
980.8895
980.57336
980.1007
979.7846
979.3119
978.9958
978.52313
978.207
977.7343
977.4182
976.9455
976.62946
976.15674
975.8406
975.36786
975.0518
974.57904
974.26306
973.7904
973.4742
973.00146
```

```
972.68536
972.21265
971.8966
971.4238
971.1078
970.635
970.319
969.8462
969.5303
969.0575
968.7414
968.2687
967.9527
967.47986
967.1638
966.6911
966.375
965.9022
965.5862
965.11346
964.7976
964.3246
964.00867
963.5358
963.2198
962.747
962.431
961.9582
961.6422
961.16943
960.8534
960.38055
960.06476
959.5918
959.2758
958.803
958.487
958.01416
957.6982
957.22534
956.9095
956.4365
956.1207
955.6478
955.3319
954.8589
954.543
954.0702
953.7542
953.28125
952.9656
952.4925
952.1767
951.7038
951.388
950.91486
950.5992
950.12616
949.81036
949.3374
949.0214
948.5485
948.2326
947.75977
947.4439
```

```
946.97095
946.6552
946.18207
945.8664
945.3933
945.0776
944.60443
944.2888
943.8157
943.49994
943.02686
942.7112
942.2381
941.9224
941.4493
941.13354
940.66046
940.3448
939.87164
939.556
939.0828
938.7672
938.294
937.9784
937.5052
937.18964
936.7165
936.4008
935.92755
935.612
935.13885
934.8232
934.35004
934.0344
933.5612
933.2456
932.7724
932.4568
931.98364
931.668
931.1948
930.8792
930.40607
930.0904
929.6172
929.30164
928.82837
928.5128
928.03955
927.724
927.2506
926.93524
926.462
926.1464
925.67316
925.3576
924.88446
924.56885
924.0955
923.78
923.3068
922.9912
922.5179
922.2025
921.7292
```

```
921.41364
920.94025
920.6248
920.1515
919.836
919.3626
919.04724
918.57385
918.2584
917.78516
917.4696
916.9962
916.68085
916.2075
915.892
915.4186
915.1032
914.6299
914.31445
913.8411
913.52563
913.05225
912.7368
912.2635
911.94806
911.4746
911.15924
910.68585
910.3704
909.89703
909.5818
909.1082
908.7929
908.31934
908.004
907.53076
907.2154
906.7418
906.42645
905.953
905.63776
905.1642
904.8488
904.37537
904.06024
903.58655
903.27124
902.7977
902.48254
902.00903
901.69366
901.2202
900.9049
900.4314
900.1162
899.6426
899.32733
898.85376
898.5385
898.065
897.74976
897.2762
896.96094
896.48737
896.1721
```

```
895.69855
895.3834
894.9097
894.59454
894.121
893.8057
893.33215
893.017
892.54333
892.2282
891.7545
891.43945
890.96564
890.65045
890.1769
889.86176
889.38806
889.07306
888.59937
888.2841
887.8104
887.4954
887.02167
886.70667
886.23285
885.91785
885.44415
885.129
884.6552
884.3402
883.8665
883.55145
883.0776
882.7626
882.2889
881.9738
881.50006
881.18506
880.71124
880.39624
879.9225
879.6074
879.1338
878.81866
878.34485
878.02985
877.5561
877.24097
876.7672
876.4523
875.9784
875.66345
875.1896
874.87463
874.4008
874.0859
873.61194
873.29706
872.8232
872.50824
872.0345
871.7195
871.24554
870.93066
870.45685
```

```
870.14185
869.6679
869.353
868.8792
868.5643
868.0903
867.77545
867.3016
866.98663
866.51276
866.1979
865.72394
865.40906
864.9352
864.62024
864.14636
863.8315
863.35754
863.04266
862.5688
862.25385
861.7799
861.4651
860.99115
860.6763
860.2023
859.88745
859.41345
859.0987
858.62463
858.30994
857.8359
857.5212
857.04706
856.73224
856.25824
855.94354
855.4695
855.15466
854.68066
854.36584
853.89185
853.5771
853.103
852.78827
852.3142
851.9996
851.5255
851.2107
850.7367
850.42194
849.9479
849.63306
849.15894
848.8443
848.37024
848.05554
847.5814
847.2668
846.7926
846.47797
846.0038
845.6893
845.21497
844.90045
```

```
844.4262
844.1115
843.6374
843.32275
842.8486
842.534
842.05975
841.7452
841.27094
840.95636
840.4821
840.1675
839.69336
839.3788
838.9045
838.5901
838.11566
837.80115
837.3269
837.01245
836.5381
836.2237
835.7494
835.4349
834.9606
834.64606
834.17175
833.85724
833.38293
833.0685
832.5942
832.2796
831.80524
831.4908
831.01654
830.70197
830.2276
829.91327
829.4389
829.12445
828.6501
828.3356
827.86127
827.5469
827.0725
826.758
826.2836
825.9693
825.4948
825.1804
824.70605
824.39166
823.91724
823.6028
823.12836
822.81396
822.3396
822.02527
821.55084
821.2364
820.762
820.4477
819.9732
819.6588
819.1843
```

```
818.87006
818.39557
818.08136
817.6069
817.2926
816.81793
816.50366
816.0291
815.7149
815.2403
814.9261
814.4516
814.1372
813.6628
813.3485
812.87396
812.55975
812.0852
811.771
811.2963
810.9821
810.50757
810.19336
809.71875
809.4045
808.92993
808.6158
808.1412
807.8269
807.3523
807.0381
806.56354
806.24927
805.7747
805.4605
804.98584
804.6717
804.1971
803.8829
803.40826
803.0941
802.61945
802.3053
801.8306
801.5166
801.04193
800.7278
800.2531
799.93896
799.46423
799.1502
798.6754
798.3614
797.8866
797.5726
797.0978
796.7838
796.30896
795.995
795.52026
795.2061
794.73145
794.4174
793.9426
793.6286
```

```
793.1539
792.8398
792.36505
792.051
791.57623
791.2622
790.7874
790.4734
789.9986
789.68463
789.2098
788.8958
788.42096
788.107
787.6321
787.3183
786.8434
786.5294
786.05457
785.7406
785.26587
784.9518
784.4769
784.1631
783.6881
783.37427
782.8993
782.5854
782.1106
781.7967
781.3218
781.0078
780.53296
780.219
779.74414
779.4301
778.9553
778.6415
778.1665
777.8527
777.3777
777.06384
776.5889
776.2751
775.8
775.4862
775.0113
774.6975
774.22253
773.9086
773.43365
773.11993
772.6449
772.3311
771.8561
771.5423
771.06726
770.75354
770.2784
769.9647
769.4896
769.1759
768.7009
768.3871
767.912
```

```
767.5982
767.1233
766.8095
766.33435
766.0207
765.5456
765.23193
764.7569
764.4431
763.96796
763.6543
763.17926
762.86554
762.39044
762.0767
761.6016
761.2879
760.8127
760.49915
760.024
759.7103
759.2352
758.9215
758.44635
758.13275
757.6576
757.34393
756.8688
756.5551
756.07996
755.76636
755.29114
754.9776
754.5023
754.1887
753.7135
753.4
752.9247
752.61115
752.13586
751.8223
751.3471
751.0335
750.5584
750.24475
749.7695
749.45593
748.9808
748.6671
748.19183
747.87836
747.40314
747.08954
746.6143
746.3007
745.8255
745.512
745.0367
744.7232
744.2479
743.93445
743.45905
743.14557
742.6703
742.3568
```

```
741.88135
741.56805
741.09265
740.77924
740.3037
739.99036
739.515
739.20154
738.72626
738.4127
737.9374
737.624
737.1486
736.8352
736.3598
736.04645
735.571
735.2576
734.7823
734.4688
733.99335
733.68005
733.2046
732.89124
732.4158
732.1024
731.62695
731.31366
730.83813
730.5249
730.0493
729.73615
729.2606
728.94727
728.4718
728.1585
727.683
727.3696
726.89417
726.5809
726.10535
725.7921
725.3165
725.0033
724.5277
724.21454
723.7389
723.42566
722.9501
722.6369
722.1613
721.848
721.3725
721.0593
720.5837
720.27057
719.7949
719.48175
719.0061
718.6929
718.2173
717.90405
717.42847
717.11536
716.63965
```

```
716.3265
715.8508
715.5377
715.062
714.74896
714.2733
713.96014
713.48444
713.1713
712.6957
712.3825
711.90686
711.5937
711.11804
710.80493
710.32916
710.0162
709.54047
709.22736
708.75165
708.43854
707.9628
707.6498
707.174
706.86096
706.3852
706.07214
705.5964
705.2834
704.80756
704.49457
704.0188
703.70575
703.23
702.917
702.44116
702.1282
701.65234
701.33936
700.8635
700.55054
700.07477
699.7617
699.28595
698.9729
698.49713
698.18414
697.7083
697.3954
696.9195
696.6066
696.13074
695.8179
695.34186
695.029
694.5531
694.2402
693.76434
693.45135
692.9755
692.6626
692.1867
691.87384
691.3979
691.0851
```

```
690.60913
690.29626
689.8203
689.50745
689.0315
688.7187
688.2427
687.9299
687.45386
687.14105
686.66504
686.3523
685.8762
685.56335
685.08746
684.7746
684.29865
683.9858
683.5098
683.1971
682.721
682.40826
681.9322
681.61945
681.1434
680.8307
680.3546
680.0419
679.56586
679.25305
678.77704
678.4643
677.9882
677.6755
677.19934
676.88666
676.4105
676.0979
675.62177
675.3091
674.83295
674.52026
674.0442
673.7315
673.25543
672.9427
672.46655
672.1539
671.67773
671.36505
670.8889
670.5763
670.10016
669.7875
669.31134
668.9987
668.5225
668.2099
667.7337
667.42114
666.9449
666.6323
666.1561
665.8435
665.3673
```

```
665.05475
664.5785
664.266
663.7897
663.4771
663.00085
662.6883
662.2121
661.8995
661.4232
661.1107
660.6345
660.32196
659.84564
659.5332
659.0569
658.7444
658.26807
657.9556
657.4793
657.1668
656.6904
656.378
655.9017
655.5892
655.11285
654.8004
654.32404
654.01154
653.5352
653.2228
652.74646
652.43396
651.9576
651.6452
651.1688
650.8564
650.38
650.06757
649.59125
649.2788
648.80237
648.49
648.0136
647.7012
647.22485
646.9124
646.436
646.1236
645.64716
645.3348
644.85834
644.546
644.0695
643.7572
643.2807
642.96844
642.49194
642.1796
641.7031
641.39087
640.91437
640.602
640.12555
639.81323
```

```
639.33673
639.0244
638.5479
638.2356
637.75916
637.44684
636.9703
636.6581
636.1815
635.86926
635.3927
635.0805
634.6039
634.2916
633.81506
633.50287
633.0263
632.71405
632.2375
631.92523
631.44867
631.1365
630.65985
630.3477
629.87103
629.5589
629.0823
628.7701
628.29346
627.98126
627.50464
627.19244
626.7159
626.4037
625.92706
625.6149
625.13824
624.8261
624.3494
624.0373
623.5606
623.24854
622.7718
622.4597
621.98303
621.67084
621.19415
620.8821
620.4054
620.0933
619.6166
619.30444
618.8278
618.5157
618.039
617.7269
617.2502
616.93805
616.46136
616.1493
615.67255
615.36053
614.8837
614.5718
614.095
```

```
613.7829
613.3061
612.99414
612.5174
612.2054
611.7285
611.41656
610.9397
610.62775
610.1509
609.839
609.36206
609.0502
608.57324
608.26135
607.7844
607.4726
606.9957
606.6837
606.2069
605.8949
605.4181
605.1062
604.6293
604.3174
603.84045
603.52856
603.05164
602.73975
602.2629
601.951
601.47406
601.16223
600.68524
600.3734
599.8964
599.5846
599.1076
598.7958
598.31885
598.00696
597.53
597.2182
596.7412
596.42944
595.9524
595.6406
595.1636
594.8518
594.37476
594.06305
593.586
593.27423
592.7972
592.4854
592.00836
591.69653
591.2196
590.90784
590.4308
590.119
589.64197
589.33026
588.85315
588.54144
```

```
588.0644
587.7526
587.2756
586.9638
586.48676
586.17505
585.69794
585.3862
584.9092
584.5974
584.1203
583.80865
583.3315
583.01984
582.5427
582.231
581.7539
581.44226
580.9651
580.6535
580.17633
579.8647
579.3875
579.07587
578.5987
578.28705
577.8099
577.4982
577.02106
576.7095
576.2323
575.9207
575.4434
575.13196
574.65466
574.3431
573.86584
573.5543
573.0771
572.7655
572.28827
571.9766
571.49945
571.18787
570.71063
570.3992
569.9218
569.61035
569.133
568.82153
568.34424
568.0327
567.5554
567.24396
566.7666
566.45514
565.9778
565.6663
565.18896
564.8775
564.4002
564.08875
563.6114
563.2999
562.8226
```

```
562.51117
562.03375
561.72235
561.24493
560.93353
560.4561
560.1448
559.66736
559.35596
558.8786
558.56714
558.0898
557.7784
557.30096
556.98956
556.51215
556.20074
555.72327
555.4119
554.9345
554.62317
554.1457
553.83435
553.35693
553.0456
552.5682
552.25684
551.7793
551.468
550.9905
550.67926
550.2017
549.8904
549.41284
549.1016
548.6241
548.3128
547.83527
547.52405
547.04645
546.73517
546.2577
545.9464
545.4689
545.15765
544.68005
544.36884
543.89124
543.5801
543.1024
542.7912
542.3136
542.00244
541.52484
541.2137
540.736
540.42487
539.94714
539.636
539.15845
538.8473
538.3696
538.0585
537.5808
537.26965
```

```
536.792
536.4809
536.0032
535.6921
535.2144
534.90326
534.4256
534.11444
533.63684
533.3257
532.84796
532.5369
532.05914
531.74805
531.2704
530.9593
530.48157
530.17053
529.69275
529.3817
528.90393
528.5929
528.1152
527.8041
527.3263
527.0153
526.53754
526.22656
525.7488
525.43774
524.95996
524.6489
524.17114
523.8601
523.3823
523.0713
522.59344
522.28253
521.8046
521.4938
521.01587
520.70496
520.2271
519.9162
519.4383
519.1274
518.6495
518.3385
517.86066
517.54974
517.0719
516.7609
516.283
515.97217
515.4942
515.18335
514.70544
514.3946
513.9166
513.6058
513.1278
512.81696
512.33905
512.0282
511.55026
```

```
511.23938
510.76144
510.4506
509.97263
509.66177
509.1838
508.873
508.395
508.08426
507.60617
507.29544
506.81738
506.5066
506.02856
505.71777
505.23975
504.92896
504.45102
504.14023
503.6622
503.35144
502.87332
502.5626
502.0846
501.77383
501.29578
500.98505
500.507
500.19623
499.71814
499.40738
498.92932
498.61862
498.1405
497.82983
497.3517
497.04108
496.56287
496.25223
495.7741
495.46344
494.9853
494.67465
494.19647
493.88583
493.40765
493.09705
492.6189
492.3083
491.83002
491.51944
491.0413
490.73068
490.25247
489.9419
489.46365
489.15308
488.67484
488.3643
487.8861
487.5755
487.0973
486.78668
486.30847
485.9979
```

```
485.51965
485.2091
484.7308
484.4203
483.94205
483.63144
483.15323
482.84268
482.36444
482.0539
481.57562
481.2651
480.7868
480.4763
479.998
479.68753
479.2092
478.8987
478.42038
478.1099
477.63156
477.32114
476.84274
476.53232
476.054
475.7435
475.26517
474.9547
474.47635
474.1659
473.68756
473.37714
472.89877
472.58832
472.10995
471.7995
471.3211
471.0107
470.53232
470.22183
469.7435
469.433
468.95465
468.64432
468.16586
467.8555
467.37704
467.0667
466.58823
466.27792
465.7994
465.4891
465.0106
464.70023
464.22177
463.91144
463.43295
463.12262
462.64413
462.33383
461.85532
461.54498
461.0665
460.75623
460.27768
```

```
459.96744
459.48886
459.17856
458.7
458.38974
457.9112
457.60095
457.12238
456.81213
456.33356
456.02335
455.54474
455.23456
454.75592
454.44574
453.9671
453.65695
453.17828
452.86807
452.38943
452.07925
451.60065
451.2905
450.81183
450.50168
450.023
449.71292
449.2342
448.92407
448.44537
448.13528
447.65656
447.34637
446.86774
446.5576
446.07892
445.7688
445.2901
444.98004
444.50128
444.1912
443.71246
443.40237
442.9236
442.6136
442.1348
441.82474
441.34598
441.0359
440.55716
440.2471
439.76834
439.4583
438.97958
438.66956
438.1907
437.8807
437.4019
437.09192
436.61313
436.3031
435.82428
435.5143
435.03546
434.72543
```

```
434.24664
433.9366
433.45782
433.14786
432.669
432.35907
431.8802
431.57022
431.09137
430.78143
430.30252
429.9926
429.51373
429.2038
428.7249
428.41504
427.93613
427.62613
427.14722
426.83734
426.35846
426.04855
425.56964
425.2598
424.78085
424.47095
423.99194
423.6821
423.2032
422.8933
422.41437
422.10455
421.62558
421.31573
420.83673
420.52686
420.04785
419.73807
419.25912
418.94925
418.47025
418.16046
417.68143
417.37167
416.8926
416.58286
416.10385
415.79407
415.31503
415.00522
414.5262
414.21643
413.73734
413.4276
412.94855
412.6388
412.15973
411.84998
411.3709
411.0612
410.58206
410.27237
409.79327
409.48358
409.00446
```

```
408.6948
408.21564
407.90588
407.42682
407.1171
406.63797
406.3283
405.84918
405.53955
405.06036
404.7507
404.27155
403.96185
403.48273
403.17307
402.6939
402.38422
401.9051
401.59543
401.11627
400.8066
400.3274
400.01785
399.53864
399.22903
398.74982
398.44028
397.961
397.65143
397.17215
396.86258
396.38333
396.0738
395.5945
395.28494
394.8057
394.49622
394.01688
393.7074
393.22803
392.91855
392.43924
392.12973
391.65042
391.34094
390.8616
390.55215
390.07278
389.7633
389.28397
388.9745
388.49515
388.1857
387.70636
387.39685
386.9175
386.60806
386.1287
385.81924
385.33987
385.03043
384.5511
384.2416
383.76224
383.45282
```

```
382.97342
382.66403
382.1846
381.8752
381.39575
381.08643
380.60693
380.29758
379.8181
379.50873
379.0293
378.71994
378.24048
377.93112
377.45166
377.14233
376.66284
376.35355
375.87402
375.56473
375.0852
374.77594
374.2964
373.98715
373.50757
373.1983
372.71875
372.4095
371.92993
371.6207
371.1411
370.83185
370.3523
370.0431
369.56348
369.2543
368.7747
368.46545
367.98584
367.67664
367.19705
366.88785
366.4082
366.09903
365.61942
365.31018
364.8306
364.52142
364.04178
363.73257
363.25296
362.9438
362.46414
362.15497
361.67532
361.36618
360.8865
360.57733
360.0977
359.78854
359.30884
358.99973
358.52005
358.21094
357.73126
```

```
357.42215
356.94244
356.63333
356.15363
355.84454
355.3648
355.05573
354.576
354.2669
353.7872
353.4781
352.99835
352.6893
352.20956
351.90054
351.42075
351.11176
350.63193
350.3229
349.8431
349.5341
349.05432
348.7453
348.2655
347.95648
347.47668
347.1677
346.68787
346.3789
345.89905
345.59012
345.11026
344.8013
344.32144
344.01245
343.53262
343.22366
342.7438
342.43488
341.955
341.64606
341.16623
340.85724
340.37738
340.06845
339.58856
339.27966
338.79974
338.49088
338.01093
337.70203
337.2221
336.91324
336.43332
336.12442
335.6445
335.33563
334.8557
334.54684
334.0669
333.75803
333.27805
332.9692
332.48923
332.18042
```

```
331.70044
331.39163
330.91162
330.6028
330.1228
329.81403
329.334
329.0252
328.5452
328.23642
327.75638
327.4476
326.9676
326.65878
326.17874
325.87003
325.38995
325.0812
324.60114
324.2924
323.81232
323.5036
323.0235
322.7148
322.2347
321.926
321.44586
321.13718
320.65707
320.3484
319.86826
319.55957
319.07944
318.77075
318.29065
317.98196
317.5018
317.19318
316.713
316.40436
315.9242
315.61557
315.13538
314.82675
314.34656
314.03793
313.55777
313.2491
312.76895
312.46033
311.98013
311.6715
311.1913
310.88272
310.4025
310.09393
309.6137
309.3051
308.8249
308.5163
308.03607
307.72754
307.24725
306.93872
306.45844
```

```
306.14993
305.66965
305.3611
304.88086
304.5723
304.092
303.78348
303.30322
302.9947
302.5144
302.2059
301.7256
301.41708
300.93674
300.62823
300.14792
299.83945
299.3591
299.05066
298.57028
298.26184
297.78146
297.47305
296.99268
296.68427
296.20383
295.89545
295.41504
295.1066
294.62622
294.3178
293.8374
293.52902
293.04858
292.7402
292.25977
291.95142
291.47095
291.16257
290.68213
290.37378
289.8933
289.58496
289.1045
288.79617
288.31573
288.0074
287.5269
287.21857
286.73807
286.42972
285.94925
285.64093
285.16043
284.85214
284.3716
284.06332
283.5828
283.27454
282.794
282.48572
282.00516
281.69693
281.21634
280.90808
```

```
280.42752
280.1193
279.63873
279.3305
278.84988
278.5417
278.0611
277.7529
277.27228
276.9641
276.48346
276.17526
275.69464
275.38644
274.90585
274.59766
274.117
273.80887
273.32822
273.02005
272.5394
272.2312
271.75058
271.44244
270.96176
270.65363
270.17297
269.86484
269.38412
269.07602
268.59534
268.28723
267.80655
267.4984
267.0177
266.70963
266.22888
265.9208
265.44006
265.132
264.65125
264.34317
263.86246
263.5544
263.07364
262.7656
262.28482
261.97678
261.496
261.188
260.7072
260.39917
259.91837
259.61035
259.12955
258.8216
258.34073
258.03275
257.5519
257.24396
256.76315
256.45514
255.97432
255.66635
255.18549
```

```
254.87752
254.39668
254.08875
253.60786
253.2999
252.81905
252.51111
252.03024
251.72232
251.24142
250.93353
250.45262
250.14471
249.66379
249.35588
248.87497
248.56711
248.08615
247.77829
247.29738
246.98947
246.50853
246.20067
245.71971
245.41187
244.9309
244.62305
244.14212
243.83427
243.35329
243.04547
242.56447
242.25665
241.77565
241.46783
240.98683
240.67905
240.19801
239.89023
239.40921
239.10141
238.62039
238.31264
237.8316
237.52383
237.04279
236.73503
236.25395
235.9462
235.46513
235.1574
234.67632
234.36859
233.88753
233.57979
233.09868
232.791
232.30988
232.0022
231.52109
231.21338
230.73228
230.42458
229.94344
229.63577
```

```
229.15465
228.84697
228.3658
228.05818
227.57698
227.26935
226.7882
226.48056
225.99939
225.69176
225.21057
224.90294
224.42175
224.11412
223.63293
223.32536
222.8441
222.53654
222.05528
221.74771
221.2665
220.95892
220.47769
220.17012
219.68887
219.38135
218.90005
218.59248
218.11124
217.80373
217.32243
217.01488
216.5336
216.22607
215.74477
215.43729
214.95598
214.64848
214.16716
213.85968
213.37834
213.07088
212.58954
212.28206
211.80069
211.49324
211.0119
210.70444
210.22308
209.91562
209.43427
209.1268
208.64543
208.33804
207.85666
207.5492
207.0678
206.7604
206.279
205.9716
205.4902
205.1828
204.70139
204.39397
203.91257
```

```
203.60516
203.12375
202.81639
202.33492
202.02756
201.5461
201.23877
200.75728
200.44997
199.96846
199.66116
199.17964
198.87234
198.39084
198.08353
197.60205
197.29472
196.8132
196.50592
196.02441
195.7171
195.23558
194.92833
194.44676
194.13948
193.65797
193.35071
192.86916
192.56189
192.08032
191.77309
191.2915
190.98428
190.50272
190.1955
189.7139
189.4067
188.9251
188.61789
188.13625
187.82907
187.34743
187.04025
186.55865
186.25146
185.76984
185.46268
184.98102
184.67387
184.1922
183.88506
183.4034
183.09625
182.61458
182.30745
181.82578
181.51865
181.03696
180.72984
180.24814
179.94106
179.45935
179.15224
178.67052
178.36343
```

```
177.88173
177.57465
177.09291
176.78583
176.3041
175.99701
175.51527
175.20822
174.72647
174.41942
173.93765
173.63062
173.14885
172.8418
172.36003
172.05301
171.57121
171.2642
170.7824
170.4754
169.99359
169.6866
169.20477
168.8978
168.41597
168.10901
167.62715
167.32018
166.83833
166.53139
166.04953
165.7426
165.26071
164.95378
164.47191
164.16498
163.68309
163.37617
162.89427
162.58736
162.10545
161.79857
161.31667
161.00977
160.52785
160.22095
159.73901
159.43214
158.95021
158.64336
158.1614
157.85455
157.37259
157.06575
156.58379
156.27693
155.79497
155.48814
155.00616
154.69933
154.21733
153.91052
153.42853
153.12172
152.63972
```

```
152.33293
151.8509
151.54413
151.06209
150.75533
150.27328
149.9665
149.48447
149.17772
148.69566
148.3889
147.90686
147.60011
147.11803
146.8113
146.32922
146.02248
145.54042
145.23369
144.7516
144.4449
143.96278
143.65608
143.17398
142.8673
142.38516
142.07849
141.59634
141.28969
140.80754
140.50087
140.01874
139.71207
139.22992
138.92326
138.4411
138.13446
137.65228
137.34566
136.86346
136.55685
136.07466
135.76804
135.28584
134.97923
134.49701
134.19041
133.70819
133.40163
132.91936
132.6128
132.13057
131.82399
131.34174
131.03519
130.55293
130.24637
129.7641
129.45757
128.97528
128.66876
128.18646
127.87994
127.397644
127.09114
```

```
126.608826
126.302315
125.82001
125.51352
125.03119
124.72471
124.24238
123.935875
123.453545
123.14708
122.66473
122.35828
121.875916
121.56946
121.08709
120.780655
120.29827
119.991844
119.509445
119.20303
118.72064
118.414215
117.931816
117.62542
117.143
116.83659
116.35418
116.0478
115.56537
115.25897
114.77655
114.47018
113.98773
113.68137
113.19891
112.89257
112.41023
112.10382
111.6216
111.31509
110.83308
110.52642
110.04459
109.73778
109.25636
108.94935
108.46842
108.16175
107.68078
107.37444
106.893295
106.58741
106.10668
105.80124
105.32195
105.01675
104.53869
104.23297
103.75759
103.451096
102.97931
102.672646
102.20416
101.89749
101.43238
```

```
101.125275
100.66474
100.35659
99.90226
99.59195
99.14504
98.832275
98.393036
98.07767
97.646
97.32872
96.90402
96.58469
96.16599
95.84486
95.43227
95.1085
94.70406
94.37755
93.97975
93.65116
93.25929
92.92813
92.542595
92.209076
91.83025
91.494835
91.12205
90.783775
90.41868
90.07703
89.72002
89.37445
89.025795
88.67634
88.33628
87.9838
87.651405
87.29613
86.97213
86.613754
86.29816
85.93542
85.63004
85.26299
84.96788
84.595726
84.31137
83.9343
83.66111
83.279854
83.01714
82.6313
82.38103
81.99188
81.75145
81.35964
81.12821
80.73575
80.511856
80.11807
79.902176
79.50586
79.29885
78.89929
```

```
78.70168
78.29833
78.11162
77.70507
77.529396
77.11972
76.95341
76.540535
76.38245
75.96534
75.8175
75.395164
75.25868
74.830696
74.704155
74.271484
74.154175
73.71761
73.60844
73.16786
73.06665
72.622826
72.52794
72.081505
71.993355
71.54318
71.46316
71.00843
70.93701
70.47847
70.41404
69.952644
69.89512
69.43067
69.38009
68.91218
68.8679
68.39548
68.35893
67.88116
67.85386
67.370674
67.35092
66.862625
66.84999
66.357544
66.35107
65.854935
65.85448
65.35483
65.359314
64.8563
64.86611
64.359505
64.374825
63.86506
63.885155
63.371986
63.39757
62.88058
62.9113
62.390366
62.427116
61.902058
61.944798
```

```
61.41559
61.463806
60.930874
60.98416
60.44732
60.506145
59.965523
60.029385
59.48499
59.55391
59.005768
59.07998
58.527885
58.60718
58.051617
58.135532
57.57711
57.665432
57.10434
57.19691
56.63294
56.730633
56.16395
56.266212
55.696136
55.803272
55.22963
55.34176
54.76487
54.88168
54.301468
54.42272
53.839466
53.965572
53.378544
53.5114
52.920338
53.05929
52.46413
52.609154
52.009964
52.161396
51.557793
51.71689
51.10821
51.27544
50.66105
50.837524
50.216785
50.40252
49.77589
49.970345
49.337967
49.54192
48.90431
49.117073
48.47386
48.695526
48.04632
48.27727
47.62158
47.862423
47.200283
47.451763
46.783527
```

```
47.045074
46.371716
46.64182
45.963875
46.242596
45.560333
45.848873
45.16144
45.46019
44.768528
45.076538
44.379784
44.69811
43.996433
44.323868
43.61729
43.953663
43.242012
43.587585
42.870785
43.224506
42.502804
42.86547
42.139008
42.511387
41.780384
42.162827
41.427166
41.81928
41.079285
41.48104
40.737896
41.14793
40.40184
40.820236
40.07073
40.498344
39.74453
40.182446
39.424583
39.87183
39.109737
39.56733
38.801907
39.268177
38.49941
38.974133
38.202
38.685593
37.91058
38.402912
37.625736
38.126575
37.347847
37.856018
37.07672
37.590767
36.8109
37.33103
36.54963
37.077446
36.29506
36.828915
36.045723
36.58589
```

```
35.801838
36.34733
35.562458
36.114487
35.328793
35.887177
35.100376
35.66545
34.877064
35.448868
34.658962
35.236694
34.44519
35.030193
34.23688
34.82912
34.033142
34.63402
33.835365
34.443905
33.642338
34.25847
33.45441
34.07825
33.27136
33.903156
33.0932
33.733128
32.919186
33.567715
32.749428
33.40727
32.584164
33.251717
32.424232
33.100357
32.26924
32.952873
32.11769
32.810547
31.970335
32.67226
31.827269
32.53764
31.688276
32.40712
31.553782
32.280598
31.424046
32.157932
31.298498
32.038876
31.17658
31.923405
31.05855
31.81079
30.943949
31.701765
30.833002
31.596571
30.725916
31.494896
30.622854
31.39606
30.523306
```

```
31.300182
30.426538
31.207325
30.332756
31.11715
30.24136
31.029955
30.15377
30.946037
30.069935
30.864742
29.988983
30.786234
29.910784
30.710737
29.835098
30.637545
29.762058
30.566792
29.691116
30.498676
29.62276
30.433117
29.557268
30.370226
29.494366
30.309526
29.43395
30.25076
29.375399
30.19387
29.31913
30.138647
29.264593
30.085245
29.21227
30.033718
29.162195
29.983576
29.113495
29.934965
29.066418
29.88809
29.021408
29.843283
28.978266
29.800182
28.93675
29.758398
28.896605
29.718096
28.85855
29.67905
28.822002
29.641397
28.786823
29.6049
28.7531
29.569649
28.720703
29.535774
28.689558
29.503084
28.659822
29.4714
```

```
28.631237
29.440807
28.603497
29.41138
28.576803
29.383121
28.551226
29.356049
28.526525
29.330105
28.502972
29.305225
28.480656
29.281395
28.459314
29.25848
28.438728
29.236315
28.418987
29.21485
28.400013
29.194262
28.38188
29.17447
28.364641
29.155537
28.347937
29.137297
28.331856
29.119778
28.316374
29.103
28.301628
29.086962
28.287592
29.071508
28.274221
29.056677
28.261395
29.042524
28.249006
29.028936
28.237162
29.015882
28.225807
29.003489
28.214945
28.991615
28.204475
28.980251
28.194424
28.969408
28.184864
28.95896
28.175806
28.948912
28.167154
28.939255
28.158846
28.92991
28.150824
28.920895
28.143103
28.912207
28.135593
```

```
28.903814
28.128387
28.895721
28.121471
28.88796
28.114779
28.880476
28.108322
28.873299
28.102161
28.866388
28.09626
28.859743
28.090721
28.853285
28.085424
28.847158
28.080423
28.841282
28.075628
28.8356
28.0711
28.830076
28.066706
28.824783
28.062527
28.819674
28.05853
28.814714
28.054665
28.80994
28.05092
28.805315
28.047352
28.800821
28.043919
28.796495
28.040632
28.792372
28.037556
28.788404
28.03463
28.784576
28.031843
28.780891
28.02923
28.777311
28.026735
28.773872
28.024351
28.770586
28.02206
28.767397
28.01983
28.764317
28.017689
28.761374
28.01568
28.758541
28.013792
28.755842
28.012014
28.753273
28.010355
28.750841
```

```
28.008806
28.748545
28.007341
28.746332
28.005953
28.744188
28.004631
28.74212
28.003387
28.740107
28.00218
28.738182
28.001032
28.736311
27.999912
28.734505
27.998865
28.732761
27.997849
28.731096
27.996872
28.729486
27.99594
28.727938
27.995079
28.726454
27.994316
28.725035
27.99362
28.723671
27.992989
28.722345
27.992424
28.721079
27.991892
28.719868
27.991428
28.71871
27.990997
28.717604
27.990597
28.716524
27.990217
28.715467
27.989883
28.714441
27.989603
28.71345
27.989363
28.71251
27.989159
28.711596
27.988977
28.710712
27.988815
28.709873
27.988668
28.709076
27.988548
28.708313
27.988457
28.707577
27.988401
28.706858
27.988361
```

```
28.706177
27.98833
28.705515
27.988314
28.704859
27.988295
28.704226
27.988283
28.703615
27.988287
28.703047
27.988287
28.702503
27.988287
28.701984
27.988298
28.701487
27.988314
28.701027
27.988337
28.700571
27.988375
28.70013
27.988413
28.699707
27.988451
28.699305
27.98849
28.698927
27.988523
28.69855
27.988556
28.698183
27.988588
28.69784
27.988638
28.697512
27.988688
28.697193
27.988743
28.696894
27.988802
28.696615
27.988869
28.696344
27.988934
28.696085
27.989006
28.695827
27.989065
28.695587
27.989138
28.695356
27.989202
28.695147
27.989275
28.69495
27.989353
28.694761
27.989428
28.694578
27.989508
28.694395
27.989592
28.694225
```

```
27.989676
28.69405
27.989763
28.693886
27.989845
28.693727
27.989933
28.69357
27.99001
28.693419
27.990091
28.693272
27.990177
28.693127
27.990267
28.692982
27.990355
28.692848
27.990437
28.69271
27.99052
28.69258
27.990608
28.692457
27.990696
28.692339
27.990778
28.69222
27.99086
28.692106
27.990944
28.691992
27.99103
28.69188
27.991117
28.691776
27.991205
28.691671
27.991293
28.691574
27.991385
28.691483
27.991463
28.691393
27.991549
28.69131
27.991629
28.691227
27.991705
28.691147
27.991776
28.691065
27.99185
28.690989
27.991924
28.690918
27.991995
28.69085
27.992073
28.690786
27.992144
28.69072
27.992212
28.69066
27.992281
```

```
28.690605
27.992344
28.690548
27.992407
28.690493
27.992474
28.69044
27.992533
28.690382
27.992594
28.690332
27.992653
28.690283
27.99272
28.690233
27.992775
28.69019
27.992838
28.690142
27.992895
28.690092
27.992958
28.69005
27.99301
28.690006
27.99307
28.689962
27.99312
28.689917
27.993172
28.689878
27.993225
28.689835
27.993275
28.689796
27.993319
28.689758
27.993366
28.689716
27.99341
28.689678
27.993464
28.689638
27.993507
28.689602
27.993553
28.689564
27.993603
28.68953
27.993647
28.68949
27.99369
28.689455
27.993729
28.689423
27.993776
28.689386
27.993816
28.689354
27.993858
28.689316
27.993898
28.689285
27.993938
28.689249
```

```
27.993967
28.689217
27.994009
28.689186
27.99404
28.689157
27.994066
28.689129
27.994102
28.689096
27.99413
28.68907
27.994158
28.689037
27.994184
28.689009
27.994215
28.688982
27.994234
28.688955
27.994255
28.68893
27.994278
28.688902
27.994299
28.688875
27.994322
28.68885
27.99434
28.688824
27.994358
28.688795
27.994377
28.68877
27.994396
28.688747
27.99441
28.688723
27.994429
28.688694
27.994442
28.688673
27.994453
28.688646
27.99446
28.688623
27.99447
28.688599
27.994474
28.688578
27.99448
28.688549
27.994486
28.68853
27.994495
28.688505
27.994503
28.688484
27.994505
28.688457
27.994509
28.688435
27.994516
28.688414
27.994524
```

```
28.68839
27.99453
28.68837
27.99454
28.688341
27.994541
28.68832
27.994547
28.6883
27.994553
28.688274
27.99456
28.68825
27.994558
28.688227
27.994566
28.688202
27.994568
28.68818
27.994568
28.688156
27.994568
28.688135
27.994572
28.68811
27.994568
28.688091
27.994568
28.688068
27.994568
28.688047
27.994562
28.688023
27.994562
28.688004
27.994558
28.68798
27.994553
28.68796
27.994549
28.687935
27.994547
28.687916
27.994541
28.687893
27.994535
28.687868
27.99453
28.687847
27.994524
28.687828
27.994518
28.687803
27.994514
28.687784
27.99451
28.68776
27.994505
28.68774
27.994497
28.68772
27.994495
28.687695
27.994486
28.68767
```

```
27.99448
28.687653
27.99447
28.687632
27.994465
28.687603
27.994453
28.687584
27.994448
28.687563
27.994436
28.68754
27.994427
28.68752
27.994421
28.6875
27.99441
28.687479
27.994396
28.687456
27.994387
28.687435
27.994377
28.687412
27.99436
28.687391
27.994352
28.687368
27.994343
28.687347
27.994333
28.687325
27.994318
28.687305
27.99431
28.687284
27.994297
28.68726
27.994286
28.687243
27.994272
28.687222
27.99426
28.687199
27.994251
28.687178
27.994234
28.687155
27.994223
28.687136
27.99421
28.68711
27.994198
28.68709
27.994183
28.687069
27.994171
28.687046
27.994158
28.687025
27.994146
28.687006
27.99413
28.686987
27.994116
```

```
28.686962
27.994102
28.686943
27.994085
28.68692
27.99408
28.6869
27.994064
28.686876
27.994047
28.686855
27.994034
28.686832
27.99402
28.686817
27.994003
28.686792
27.993992
28.68677
27.993977
28.68675
27.993965
28.68673
27.993952
28.686707
27.993935
28.686686
27.993921
28.686663
27.99391
28.686642
27.993895
28.686619
27.99388
28.686604
27.993866
28.686579
27.99385
28.68656
27.993834
28.686537
27.99382
28.686516
27.993807
28.686495
27.99379
28.686472
27.993773
28.686451
27.99376
28.686428
27.99374
28.686407
27.993727
28.686384
27.993708
28.686369
27.99369
28.686344
27.993675
28.686325
27.993658
28.686302
27.99364
28.686281
```

```
27.993622
28.686258
27.993608
28.686237
27.993591
28.686214
27.993576
28.6862
27.993559
28.686174
27.993538
28.686152
27.993519
28.68613
27.993498
28.686111
27.993479
28.686089
27.993465
28.686068
27.993444
28.686049
27.993427
28.686024
27.993406
28.686007
27.993391
28.685984
27.993372
28.68596
27.993351
28.685942
27.993334
28.68592
27.993315
28.685898
27.993296
28.685877
27.993275
28.685854
27.993254
28.685833
27.993235
28.68581
27.993214
28.685795
27.993196
28.68577
27.993176
28.685751
27.993158
28.685728
27.99314
28.685707
27.993118
28.685688
27.993095
28.685663
27.993076
28.68564
27.993053
28.685621
27.993034
28.6856
27.993015
```

```
28.685577
27.992996
28.685556
27.992975
28.685537
27.992958
28.685514
27.992937
28.685493
27.992918
28.68547
27.992895
28.685452
27.992874
28.68543
27.992857
28.68541
27.992834
28.685387
27.992815
28.685368
27.992796
28.685347
27.992775
28.685328
27.992756
28.685305
27.992733
28.68528
27.992714
28.685259
27.992693
28.68524
27.992674
28.685217
27.992653
28.685198
27.992632
28.685173
27.992613
28.685154
27.992594
28.685133
27.99257
28.68511
27.992552
28.68509
27.992533
28.68507
27.992517
28.685045
27.992495
28.685022
27.992474
28.685003
27.992455
28.684984
27.992432
28.684963
27.992413
28.68494
27.99239
28.68492
27.992374
28.6849
```

```
27.992355
28.684875
27.992327
28.684856
27.992311
28.684832
27.992289
28.684816
27.99227
28.684793
27.992249
28.684772
27.992231
28.68475
27.992207
28.684729
27.992188
28.684706
27.99217
28.684687
27.99215
28.684666
27.992132
28.68464
27.992113
28.684622
27.992088
28.684603
27.992073
28.68458
27.992054
28.684559
27.992033
28.684536
27.992012
28.684515
27.991987
28.684496
27.991968
28.684473
27.991945
28.684448
27.99193
28.684433
27.991907
28.68441
27.991892
28.68439
27.991869
28.684366
27.991854
28.684345
27.99183
28.684324
27.99181
28.684301
27.991789
28.68428
27.99177
28.684263
27.99175
28.684238
27.991726
28.68422
27.99171
```

```
28.684195
27.991682
28.684175
27.991667
28.684155
27.99165
28.684132
27.991629
28.684116
27.991611
28.684093
27.991589
28.684072
27.99157
28.68405
27.991549
28.684025
27.991526
28.684004
27.991507
28.68398
27.991488
28.683962
27.991468
28.683943
27.991447
28.683918
27.991425
28.683899
27.991405
28.683878
27.991386
28.683855
27.991365
28.683836
27.991344
28.683815
27.991323
28.683792
27.991306
28.683771
27.991285
28.68375
27.991266
28.683727
27.991243
28.683706
27.991228
28.683683
27.991205
28.683664
27.991186
28.683643
27.991165
28.683624
27.991142
28.683601
27.991125
28.68358
27.991106
28.683558
27.991087
28.683538
27.991066
28.683517
```

```
27.991043
28.683498
27.991028
28.683475
27.991005
28.683455
27.990986
28.683432
27.990965
28.68341
27.990944
28.683388
27.990921
28.683367
27.990904
28.683348
27.990881
28.683325
27.99086
28.683306
27.99084
28.683285
27.990818
28.683262
27.990803
28.68324
27.990778
28.683218
27.990755
28.683197
27.99074
28.683176
27.990715
28.683159
27.990696
28.683136
27.990673
28.683115
27.990654
28.683094
27.990633
28.683071
27.990612
28.68305
27.990593
28.683027
27.990568
28.683006
27.99055
28.682983
27.99053
28.682962
27.990509
28.682943
27.990492
28.682924
27.990467
28.682901
27.990446
28.68288
27.990427
28.682858
27.990404
28.682837
27.990383
```

```
28.682814
27.99036
28.682793
27.990345
28.68277
27.990322
28.682749
27.990301
28.682732
27.990278
28.682705
27.990261
28.682688
27.990242
28.682667
27.990217
28.682644
27.990196
28.682623
27.990173
28.682602
27.990158
28.682579
27.990135
28.682562
27.990114
28.68254
27.990095
28.68252
27.99007
28.682497
27.990053
28.682476
27.990032
28.682453
27.99001
28.682432
27.98999
28.682413
27.989971
28.682388
27.989946
28.68237
27.989927
28.682346
27.989904
28.682327
27.98989
28.682306
27.989866
28.682283
27.989845
28.682262
27.989822
28.68224
27.989803
28.68222
27.989784
28.682201
27.989763
28.682177
27.98974
28.682158
27.98972
28.682137
```

```
27.9897
28.682114
27.98968
28.682093
27.98966
28.68207
27.98964
28.682049
27.98962
28.68203
27.989597
28.682005
27.989576
28.681988
27.989553
28.681967
27.989532
28.681946
27.989515
28.681923
27.989494
28.681902
27.989471
28.681879
27.98945
28.681858
27.989428
28.681835
27.989412
28.681814
27.98939
28.681791
27.989368
28.681776
27.989347
28.681753
27.989328
28.681732
27.989307
28.68171
27.989285
28.681688
27.989265
28.68167
27.989244
28.681646
27.989225
28.681627
27.989202
28.681606
27.989183
28.681583
27.989164
28.681562
27.98914
28.68154
27.989119
28.681519
27.9891
28.681496
27.989077
28.68148
27.989058
28.681458
27.989038
```

```
28.681437
27.989016
28.681414
27.988995
28.681393
27.988976
28.681372
27.988953
28.681349
27.988934
28.681328
27.988913
28.681305
27.988895
28.681284
27.988869
28.681261
27.988852
28.68124
27.98883
28.681221
27.988808
28.681196
27.988787
28.68118
27.988764
28.681158
27.988747
28.681135
27.988726
28.681114
27.988703
28.681091
27.988682
28.68107
27.988659
28.681047
27.988638
28.681032
27.98862
28.68101
27.9886
28.680988
27.98858
28.680965
27.988562
28.680944
27.988539
28.680922
27.988518
28.6809
27.988495
28.68088
27.988474
28.68086
27.988457
28.680838
27.988436
28.680819
27.988419
28.680798
27.988394
28.680775
27.988375
28.680754
```

```
27.988352
28.68073
27.98833
28.680712
27.988308
28.68069
27.988293
28.680668
27.98827
28.680643
27.988249
28.680628
27.98823
28.680605
27.988207
28.68058
27.988188
28.680561
27.988167
28.68054
27.988144
28.680517
27.988127
28.680502
27.988106
28.680477
27.988083
28.680454
27.988068
28.680435
27.988039
28.680414
27.98802
28.680391
27.987999
28.68037
27.98798
28.680347
27.98796
28.680326
27.987942
28.680305
27.987919
28.680283
27.987898
28.680264
27.987875
28.680244
27.987856
28.680223
27.987835
28.6802
27.987816
28.68018
27.987793
28.68016
27.987774
28.680136
27.987751
28.680117
27.98773
28.680092
27.98771
28.680073
27.987688
```

```
28.680054
27.987669
28.68003
27.98765
28.68001
27.98763
28.679987
27.98761
28.679966
27.987585
28.679947
27.987566
28.679922
27.987543
28.679903
27.987524
28.67988
27.987503
28.67986
27.98748
28.67984
27.987461
28.679817
27.987438
28.679796
27.987417
28.679777
27.987398
28.679754
27.987381
28.679733
27.98736
28.679712
27.987337
28.679688
27.987316
28.67967
27.987293
28.679646
27.987272
28.679626
27.987255
28.679605
27.987234
28.679583
27.987215
28.679564
27.987194
28.679543
27.987171
28.67952
27.98715
28.679499
27.98713
28.679476
27.987106
28.679455
27.987087
28.679436
27.987068
28.679413
27.987047
28.679392
27.987024
28.679369
```

```
27.98701
28.679348
27.986984
28.679325
27.986965
28.679306
27.986942
28.679285
27.986923
28.679262
27.986904
28.679243
27.986883
28.679222
27.98686
28.679201
27.98684
28.679178
27.986822
28.67916
27.9868
28.679138
27.986778
28.679115
27.986755
28.679096
27.986734
28.679075
27.986715
28.679052
27.986696
28.679031
27.986673
28.679008
27.986652
28.67899
27.986633
28.678968
27.986614
28.678946
27.986591
28.678926
27.98657
28.678905
27.986547
28.678883
27.986528
28.678862
27.986506
28.678839
27.986486
28.678818
27.986465
28.678799
27.986443
28.678774
27.986423
28.678755
27.986403
28.678732
27.986383
28.67871
27.98636
28.678692
27.986341
```

```
28.678669
27.986322
28.67865
27.986301
28.678627
27.986279
28.678606
27.98626
28.678585
27.986238
28.678564
27.986217
28.678545
27.986198
28.67852
27.986177
28.678501
27.986158
28.678482
27.986134
28.67846
27.986115
28.678438
27.986092
28.678415
27.986073
28.678394
27.986052
28.678371
27.986032
28.678352
27.98601
28.678331
27.985989
28.678308
27.985966
28.678288
27.985947
28.678265
27.985926
28.678244
27.985907
28.678225
27.985886
28.6782
27.985865
28.678177
27.985842
28.678162
27.985823
28.678139
27.985804
28.678118
27.985783
28.678095
27.985764
28.678076
27.98574
28.678053
27.985722
28.67803
27.9857
28.67801
27.985682
28.67799
```

```
27.985659
28.677967
27.985638
28.677948
27.985615
28.677927
27.985598
28.677904
27.985577
28.677883
27.985554
28.677864
27.985533
28.677841
27.98551
28.67782
27.985495
28.677797
27.985472
28.677778
27.98545
28.677753
27.985432
28.677734
27.985409
28.677713
27.98539
28.677694
27.985369
28.67767
27.985346
28.67765
27.985325
28.677628
27.985302
28.677607
27.985285
28.677584
27.985264
28.677563
27.98524
28.677544
27.98522
28.677519
27.985197
28.677502
27.985178
28.677479
27.985159
28.67746
27.98514
28.677437
27.98512
28.677416
27.985094
28.677393
27.985077
28.677372
27.985056
28.677353
27.985037
28.67733
27.985014
28.677309
27.984995
```

```
28.677286
27.984974
28.677265
27.984951
28.677242
27.98493
28.677227
27.984913
28.677204
27.98489
28.677183
27.98487
28.67716
27.98485
28.67714
27.984827
28.677116
27.984806
28.677095
27.984787
28.677076
27.984768
28.677053
27.984743
28.677032
27.984724
28.677013
27.984705
28.67699
27.984682
28.67697
27.984661
28.676947
27.984642
28.676926
27.984621
28.676905
27.9846
28.676882
27.984577
28.676865
27.984558
28.676842
27.984539
28.676819
27.984518
28.6768
27.984499
28.676779
27.984476
28.676756
27.984455
28.676735
27.984436
28.676712
27.984417
28.676691
27.984392
28.676668
27.984373
28.676647
27.984352
28.67663
27.984331
28.676609
```

```
27.984308
28.676586
27.984293
28.676565
27.984268
28.676542
27.98425
28.676521
27.984226
28.676502
27.984207
28.676483
27.984182
28.676458
27.984163
28.67644
27.984142
28.676416
27.984123
28.676395
27.9841
28.676373
27.98408
28.676352
27.984056
28.676332
27.984041
28.676308
27.984024
28.676287
27.984
28.676268
27.98398
28.676245
27.983957
28.676226
27.983936
28.676205
27.983913
28.676182
27.983898
28.67616
27.983875
28.676138
27.983854
28.676117
27.983831
28.676094
27.983812
28.676079
27.983793
28.676054
27.983772
28.676035
27.98375
28.676012
27.983728
28.675991
27.983711
28.675968
27.983686
28.675947
27.983667
28.675928
27.983648
```

```
28.675909
27.983625
28.675886
27.983606
28.675865
27.983585
28.675842
27.983562
28.675821
27.983541
28.675798
27.983522
28.67578
27.983503
28.675758
27.98348
28.675737
27.98346
28.675714
27.983437
28.67569
27.983416
28.675669
27.983398
28.675652
27.98338
28.67563
27.983356
28.675608
27.983337
28.675587
27.983316
28.675564
27.983294
28.675543
27.983274
28.675524
27.983255
28.675505
27.983234
28.675482
27.983212
28.67546
27.983192
28.675438
27.983168
28.675417
27.983147
28.675394
27.983124
28.675373
27.983109
28.67535
27.983086
28.67533
27.983065
28.67531
27.983042
28.675291
27.983025
28.675268
27.983004
28.675247
27.98298
28.675226
```

```
27.982962
28.675203
27.982943
28.675184
27.982922
28.675165
27.982903
28.67514
27.982878
28.675121
27.982859
28.6751
27.982836
28.675077
27.982817
28.675056
27.982796
28.675034
27.982777
28.675013
27.982754
28.67499
27.982735
28.67497
27.982716
28.674946
27.982693
28.67493
27.982672
28.674908
27.982653
28.674887
27.98263
28.674864
27.982609
28.674843
27.982592
28.67482
27.98257
28.6748
27.982548
28.674776
27.982527
28.67476
27.982504
28.674738
27.982485
28.674717
27.982466
28.674694
27.982445
28.674673
27.982422
28.674652
27.982399
28.67463
27.98238
28.674608
27.98236
28.674585
27.98234
28.67457
27.982317
28.674545
27.982302
```

```
28.674526
27.982279
28.674503
27.982258
28.674482
27.982235
28.67446
27.982216
28.674438
27.982197
28.674416
27.982176
28.6744
27.982155
28.674376
27.982136
28.674356
27.982115
28.674334
27.982092
28.674313
27.98207
28.67429
27.982052
28.674269
27.982027
28.674246
27.98201
28.674227
27.981989
28.674206
27.981964
28.674181
27.981945
28.674164
27.981926
28.674143
27.981907
28.67412
27.981882
28.674099
27.981865
28.674078
27.98184
28.674055
27.981821
28.674034
27.981802
28.674015
27.98178
28.673996
27.981758
28.673973
27.981735
28.673952
27.98172
28.67393
27.981697
28.673908
27.981676
28.673885
27.981657
28.673864
27.981634
28.673845
```

```
27.981613
28.673826
27.981594
28.673798
27.981575
28.673777
27.981552
28.67376
27.981533
28.673738
27.981514
28.673716
27.98149
28.673695
27.98147
28.673676
27.981451
28.67365
27.981426
28.673628
27.981407
28.673613
27.981384
28.67359
27.981369
28.673569
27.981344
28.673546
27.981325
28.673527
27.981302
28.673506
27.981281
28.673481
27.981258
28.673462
27.98124
28.67344
27.981218
28.673418
27.9812
28.673393
27.981178
28.673374
27.981157
28.673353
27.981138
28.673334
27.98112
28.673311
27.981096
28.67329
27.981075
28.673267
27.981052
28.673248
27.981033
28.673227
27.981012
28.673204
27.980995
28.673185
27.980974
28.673164
27.980955
```

```
28.673141
27.98093
28.67312
27.980911
28.673098
27.98089
28.673077
27.98087
28.673058
27.980846
28.673035
27.980827
28.673016
27.980803
28.672995
27.980787
28.672972
27.980764
28.67295
27.980743
28.67293
27.98072
28.672907
27.980705
28.672888
27.980682
28.672863
27.980661
28.672848
27.980642
28.672825
27.980618
28.672804
27.9806
28.672781
27.98058
28.67276
27.980556
28.672737
27.980536
28.672716
27.980515
28.672697
27.980495
28.672674
27.980474
28.672653
27.980452
28.672634
27.98043
28.67261
27.980413
28.672586
27.980392
28.672567
27.980368
28.672546
27.980349
28.672523
27.980331
28.672504
27.98031
28.67248
27.980288
28.672464
```

```
27.980267
28.67244
27.980247
28.672419
27.980225
28.672396
27.980202
28.672379
27.980183
28.672358
27.980167
28.672337
27.980143
28.672312
27.98012
28.672289
27.9801
28.672268
27.980076
28.67225
27.980057
28.67223
27.980038
28.672207
27.980019
28.672188
27.979998
28.672167
27.979975
28.672142
27.979954
28.672123
27.979937
28.6721
27.979914
28.67208
27.979893
28.672056
27.979876
28.672037
27.97985
28.672016
27.979826
28.671993
27.979807
28.671974
27.979788
28.67195
27.979773
28.671928
27.979748
28.67191
27.979729
28.671886
27.979706
28.671865
27.979685
28.671844
27.979662
28.671824
27.979643
28.671803
27.979624
28.671783
27.979603
```

```
28.671759
27.97958
28.67174
27.97956
28.671719
27.979542
28.671696
27.979523
28.671675
27.9795
28.671656
27.979479
28.671633
27.979456
28.671612
27.979437
28.671589
27.979416
28.671568
27.979397
28.671549
27.979374
28.671526
27.979355
28.671505
27.979334
28.671482
27.979311
28.671463
27.979294
28.671438
27.97927
28.671423
27.97925
28.671398
27.97923
28.671375
27.979206
28.671356
27.979185
28.671335
27.979168
28.671316
27.979147
28.671291
27.979124
28.67127
27.979109
28.671247
27.979086
28.671228
27.979065
28.671207
27.979042
28.671185
27.979023
28.671165
27.979002
28.671144
27.97898
28.671122
27.97896
28.6711
27.978937
28.671082
```

```
27.978922
28.671059
27.978899
28.671038
27.978878
28.671015
27.978855
28.670994
27.978834
28.67097
27.978815
28.670952
27.978796
28.67093
27.978773
28.670908
27.978752
28.670889
27.978735
28.670864
27.978714
28.670845
27.978691
28.670826
27.97867
28.670805
27.978647
28.670782
27.978628
28.670763
27.97861
28.670742
27.978586
28.67072
27.978565
28.670698
27.978546
28.670675
27.978523
28.670654
27.978504
28.670633
27.978483
28.67061
27.978464
28.67059
27.978445
28.67057
27.97842
28.670551
27.978401
28.670527
27.978378
28.670504
27.97836
28.670485
27.97834
28.670464
27.978317
28.67044
27.978294
28.67042
27.978273
28.670397
27.978258
```

```
28.670378
27.978233
28.670359
27.978214
28.670338
27.978191
28.670313
27.978172
28.670294
27.978153
28.67027
27.978132
28.67025
27.97811
28.67023
27.978088
28.670208
27.97807
28.670189
27.97805
28.670168
27.978027
28.670143
27.978004
28.670122
27.977983
28.670103
27.977964
28.67008
27.977945
28.67006
27.977922
28.67004
27.977903
28.670017
27.977879
28.669998
27.977863
28.669977
27.97784
28.669954
27.97782
28.669933
27.977797
28.66991
27.977777
28.66989
27.977758
28.669867
27.977737
28.669847
27.977715
28.669823
27.977694
28.669807
27.977676
28.669785
27.977654
28.669764
27.977633
28.669739
27.977613
28.66972
27.977594
28.6697
```

```
27.977571
28.669678
27.97755
28.669653
27.977528
28.669634
27.977509
28.669613
27.977488
28.66959
27.977468
28.66957
27.977446
28.66955
27.977425
28.669529
27.977407
28.669506
27.977385
28.669485
27.977364
28.669466
27.977346
28.669441
27.977322
28.669422
27.977303
28.669403
27.977282
28.66938
27.977259
28.66936
27.977238
28.669336
27.97722
28.669315
27.9772
28.669292
27.977177
28.669273
27.977158
28.669252
27.977133
28.669233
27.977114
28.66921
27.977095
28.66919
27.977074
28.669167
27.977057
28.669146
27.977034
28.669123
27.977009
28.669102
27.97699
28.669083
27.976969
28.669058
27.97695
28.669039
27.976927
28.66902
27.976908
```

```
28.668997
27.976887
28.668976
27.976864
28.668955
27.976849
28.668932
27.976824
28.668911
27.976805
28.668894
27.976784
28.668867
27.97676
28.668848
27.97674
28.668829
27.97672
28.66881
27.9767
28.668785
27.976677
28.668762
27.976662
28.668741
27.976639
28.668718
27.976618
28.668697
27.976595
28.668678
27.97658
28.668655
27.976557
28.668636
27.976536
28.668615
27.976513
28.668592
27.976496
28.668571
27.976473
28.668549
27.976452
28.668528
27.97643
28.668505
27.976408
28.668484
27.976387
28.668467
27.97637
28.668442
27.976349
28.668423
27.97633
28.668402
27.976305
28.66838
27.976286
28.668358
27.976263
28.668337
27.976248
28.668318
```

```
27.976223
28.668293
27.976204
28.66827
27.976181
28.668251
27.976162
28.668232
27.976141
28.668211
27.976118
28.668188
27.976097
28.668167
27.97608
28.668144
27.976057
28.668123
27.976036
28.668104
27.976013
28.66808
27.975992
28.66806
27.975973
28.668041
27.975954
28.668018
27.975935
28.667997
27.975916
28.667974
27.975893
28.667953
27.975872
28.667936
27.975853
28.66791
27.975832
28.667892
27.975811
28.667871
27.975788
28.667849
27.975767
28.667828
27.975744
28.667807
27.975729
28.667784
27.975706
28.667763
27.975685
28.66774
27.975666
28.667719
27.975647
28.6677
27.975624
28.66768
27.975603
28.667658
27.97558
28.667637
27.975563
```

```
28.667614
27.975542
28.667593
27.975517
28.66757
27.975498
28.66755
27.975475
28.66753
27.97546
28.667511
27.975437
28.667488
27.975412
28.667467
27.975393
28.667444
27.975372
28.667423
27.975353
28.6674
27.975334
28.66738
27.975311
28.66736
27.97529
28.667337
27.975267
28.667316
27.975248
28.667292
27.97523
28.667274
27.975208
28.667252
27.975185
28.667233
27.975168
28.66721
27.975147
28.66719
27.975128
28.66717
27.975105
28.667145
27.975084
28.667126
27.975061
28.667103
27.975042
28.667084
27.975021
28.667059
27.974998
28.66704
27.97498
28.667019
27.97496
28.666996
27.97494
28.666975
27.974916
28.666952
27.974895
28.666937
```

```
27.974876
28.666914
27.974855
28.66689
27.974834
28.66687
27.974812
28.66685
27.97479
28.666826
27.974773
28.666805
27.974752
28.666786
27.97473
28.666767
27.97471
28.666744
27.97469
28.666723
27.97467
28.666697
27.974648
28.66668
27.974627
28.666658
27.974607
28.666636
27.974585
28.666615
27.974566
28.666592
27.974543
28.66657
27.974524
28.666548
27.974503
28.666533
27.97448
28.66651
27.97446
28.666489
27.974442
28.666466
27.974422
28.666447
27.974398
28.666426
27.974379
28.666403
27.974358
28.666382
27.974335
28.666359
27.974314
28.66634
27.974297
28.666319
27.974277
28.666296
27.974255
28.666275
27.974234
28.666252
27.97421
```

```
28.666233
27.974192
28.666208
27.97417
28.666193
27.974148
28.666168
27.974133
28.66615
27.97411
28.666126
27.974089
28.666103
27.974066
28.666084
27.974045
28.666061
27.974026
28.66604
27.974007
28.666023
27.973984
28.666002
27.973965
28.665977
27.973944
28.665958
27.973923
28.665936
27.973902
28.665915
27.973879
28.665895
27.973864
28.665873
27.97384
28.665852
27.97382
28.665829
27.973797
28.665808
27.973776
28.665785
27.973759
28.665766
27.973738
28.665745
27.973715
28.665722
27.973694
28.6657
27.973675
28.665678
27.973652
28.665663
27.97363
28.665636
27.97361
28.665613
27.973589
28.665596
27.973572
28.665573
27.97355
28.66555
```

```
27.973528
28.665531
27.973507
28.66551
27.973484
28.665487
27.973463
28.665468
27.973446
28.665447
27.973427
28.665424
27.973406
28.665403
27.973381
28.66538
27.973364
28.66536
27.97334
28.665337
27.97332
28.665318
27.973299
28.665297
27.973278
28.665274
27.973259
28.665253
27.973234
28.665234
27.973215
28.66521
27.973194
28.665192
27.973175
28.66517
27.973156
28.665148
27.973133
28.665127
27.973114
28.665104
27.973095
28.665085
27.973074
28.665066
27.973055
28.665041
27.973032
28.66502
27.973011
28.665
27.97299
28.66498
27.972971
28.664957
27.972946
28.664936
27.972927
28.664913
27.972908
28.664892
27.972887
28.664873
27.972864
```

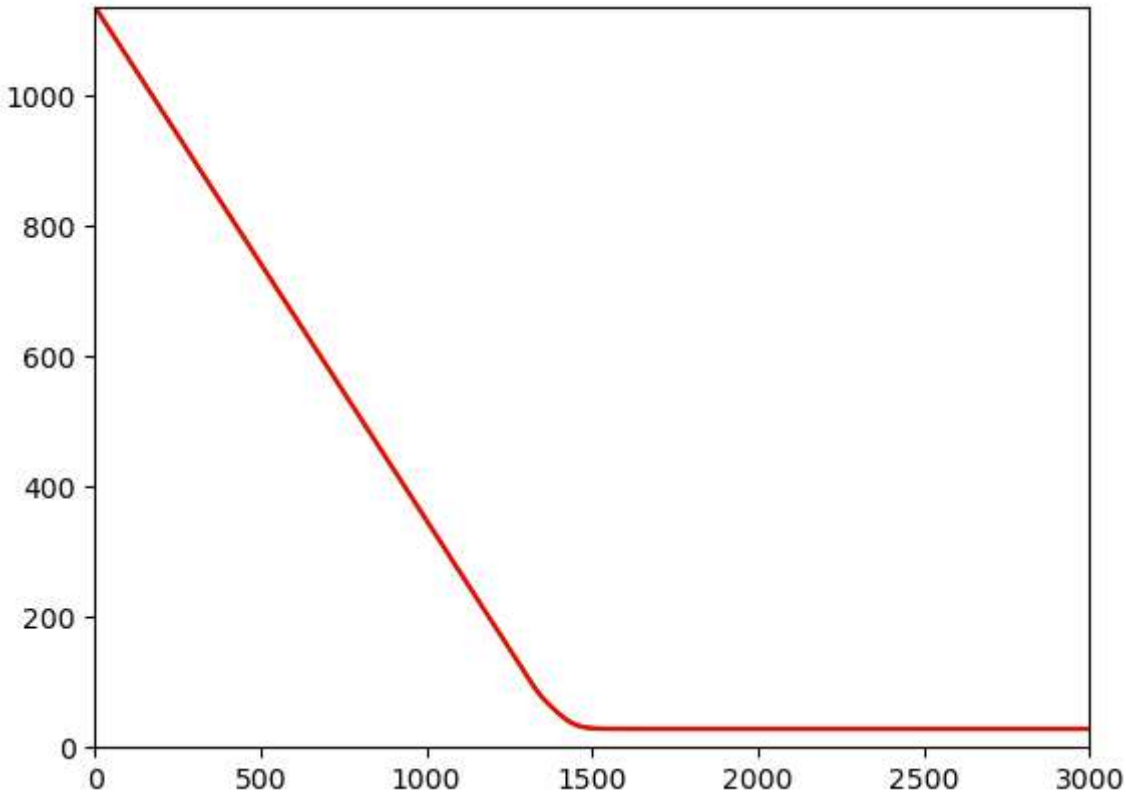
```
28.66485
27.972843
28.66483
27.972824
28.664806
27.972801
28.664787
27.97278
28.664766
27.972761
28.664743
27.972742
28.664724
27.97272
28.664703
27.9727
28.66468
27.972681
28.664661
27.972662
28.664637
27.972637
28.664616
27.972618
28.664597
27.972595
28.664577
27.97257
28.664553
27.972551
28.664534
27.972532
28.66451
27.972513
28.66449
27.972492
28.66447
27.972473
28.664448
27.972448
28.664427
27.972431
28.664406
27.97241
28.664383
27.97239
28.664362
27.972368
28.664343
27.97235
28.66432
27.972326
28.664299
27.972301
28.664276
27.972282
28.664255
27.972263
28.664236
27.972242
28.664213
27.972223
28.664192
27.9722
28.66417
```

```
27.97218
28.664148
27.972157
28.66413
27.972137
28.664106
27.972118
28.664085
27.9721
28.664062
27.972078
28.664042
27.972057
28.664022
27.972038
28.664
27.972017
28.663979
27.971998
28.66396
27.971973
28.663937
27.97195
28.663916
27.971931
28.663897
27.971912
28.663874
27.971888
28.66385
27.971872
28.663828
27.97185
28.663809
27.971828
28.663788
27.97181
28.663765
27.971786
28.663746
27.971767
28.663727
27.971746
28.663702
27.971725
28.663683
27.971706
28.663658
27.971685
28.66364
27.971663
28.663618
27.971643
28.663595
27.971619
28.663574
27.971603
28.663557
27.97158
28.66353
27.971561
28.663511
27.971537
28.663492
27.971521
```

```
28.663467
27.971497
28.663448
27.971478
28.663425
27.971455
28.663404
27.971436
28.663385
27.971413
28.663366
27.97139
28.66334
27.971373
28.66332
27.97135
28.6633
27.97133
28.663279
27.97131
28.663258
27.971292
28.663235
27.971268
28.663214
27.971249
28.66319
27.971228
28.663172
27.971209
28.66315
27.971186
28.663132
27.971167
28.663107
27.971142
28.663084
27.971123
28.663065
27.971102
28.663044
27.97108
28.663021
27.97106
28.663
27.971037
28.662981
27.971018
28.662958
27.970997
28.662937
27.970978
28.662918
27.970959
28.662895
27.970934
28.662874
27.970911
28.662855
27.970896
28.66283
27.970873
28.662811
27.970852
28.662788
```

```
27.970833
28.662766
27.970814
28.662748
27.97079
28.662725
27.97077
28.662704
27.970753
28.662683
27.97073
28.66266
27.970709
28.66264
27.970686
28.662617
27.970665
28.662596
27.970648
28.662579
27.970627
28.662554
27.970604
28.662533
27.970583
28.66251
27.970564
28.662489
27.970545
28.66247
27.970522
28.662447
27.9705
28.662426
27.970478
28.662403
27.97046
28.662388
27.97044
28.662365
27.970417
28.66234
27.970396
28.66232
27.970373
28.6623
27.970354
28.662277
27.970335
28.662256
27.970314
28.662233
27.970295
28.662212
27.970276
28.662195
27.970253
28.662174
27.970232
28.66215
27.97021
28.66213
27.970188
28.66211
27.970171
```

```
28.662086
27.970148
28.662066
27.970129
28.662043
27.970104
28.662022
27.970089
28.662004
27.970066
28.66198
27.970045
28.66196
27.970022
28.66194
27.970003
28.661917
27.969984
28.661896
27.969963
28.661873
27.96994
28.661852
27.969921
28.661833
27.9699
28.66181
27.969877
28.66179
27.969858
28.661764
27.969837
28.661741
27.969816
28.661726
27.969797
28.661703
27.969772
28.661682
27.969753
28.66166
27.969732
28.661638
27.969713
28.66162
27.969694
28.661598
27.969671
28.661577
27.969652
28.661556
27.969631
28.661535
27.969612
28.661512
27.969587
```



In [ ]: