Fate/Grand Order Servants Dataset Analysis and Visualization

Data Download

In [1]:
```
import requests

url = "https://api.atlasacademy.io/export/JP/nice_servant_lore_lang_en.json"

res = requests.request("GET", url)

rawData: list[dict] = res.json()

for s in rawData:
    s['traits'] = list(map(lambda x: x['name'], s['traits']))
    s['saberface'] = 'saberface' in s['traits']

f"There are {len(rawData)} servants with these columns: {list(rawData[0].keys())}"
```

Out[1]:
```
"There are 385 servants with these columns: ['id', 'collectionNo', 'name', 'origin
alName', 'ruby', 'battleName', 'originalBattleName', 'classId', 'className', 'typ
e', 'flag', 'rarity', 'cost', 'lvMax', 'extraAssets', 'gender', 'attribute', 'trai
ts', 'starAbsorb', 'starGen', 'instantDeathChance', 'cards', 'hitsDistribution',
'cardDetails', 'atkBase', 'atkMax', 'hpBase', 'hpMax', 'relateQuestIds', 'trialQue
stIds', 'growthCurve', 'atkGrowth', 'hpGrowth', 'bondGrowth', 'expGrowth', 'expFee
d', 'bondEquip', 'valentineEquip', 'valentineScript', 'ascensionAdd', 'traitAdd',
'svtChange', 'ascensionImage', 'ascensionMaterials', 'skillMaterials', 'appendSkil
lMaterials', 'costumeMaterials', 'coin', 'script', 'skills', 'classPassive', 'extr
aPassive', 'appendPassive', 'noblePhantasms', 'profile', 'saberface']"
```

Data Cleaning

Some columns are not useful for the analysis so needs to be removed to improve performances

In [2]:
```
import pandas as pd
import numpy as np
df = pd.DataFrame(rawData)

df['illustrator'] = df['profile'].map(lambda x: x['illustrator'])
df = df[df["type"] != 'enemyCollectionDetail']

cleanData = df.drop(columns=[
    'ruby',
    'originalBattleName',
    'originalName',
    'classId',
    'cost',
    'lvMax',
```

```
    'extraAssets',
    'starAbsorb',
    'starGen',
    'instantDeathChance',
    'cardDetails',
    'relateQuestIds',
    'trialQuestIds',
    'valentineEquip',
    'valentineScript',
    'ascensionAdd',
    'traitAdd',
    'svtChange',
    'ascensionImage',
    'coin',
    'script',
    'profile'
])

cleanData.filter(items=['name', 'className', 'type', 'rarity', 'gender', 'traits' '
```

Out[2]:

	name	className	type	rarity	gender
0	Altria Pendragon	saber	normal	5	female
1	Altria Pendragon (Alter)	saber	normal	4	female
2	Altria Pendragon (Lily)	saber	normal	4	female
3	Nero Claudius	saber	normal	4	female
4	Nero Claudius (Bride)	saber	normal	5	female
...	...	...	...	...	...
374	Hephaestion	pretender	normal	4	female
375	Lady Avalon	pretender	normal	5	female
376	Nine-Tattoo Dragon Eliza	pretender	normal	4	female
377	Tenochtitlan	pretender	normal	4	female
378	Sodom's Beast/Draco	beast	normal	5	female

378 rows × 5 columns

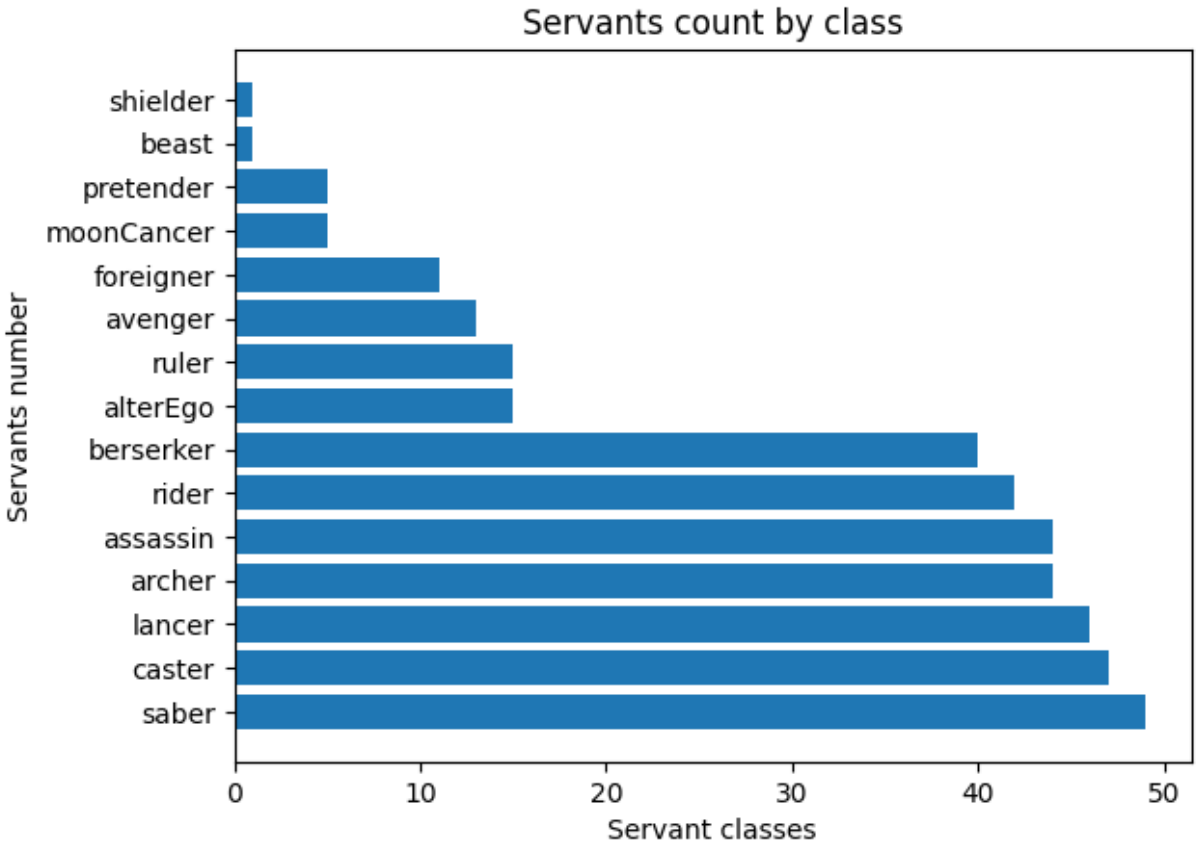
Data Caching

In [3]:
```
cleanData.to_json("dataset.json", indent=1)
```

In [4]:
```
import matplotlib.pyplot as plt
```

Servants per Class

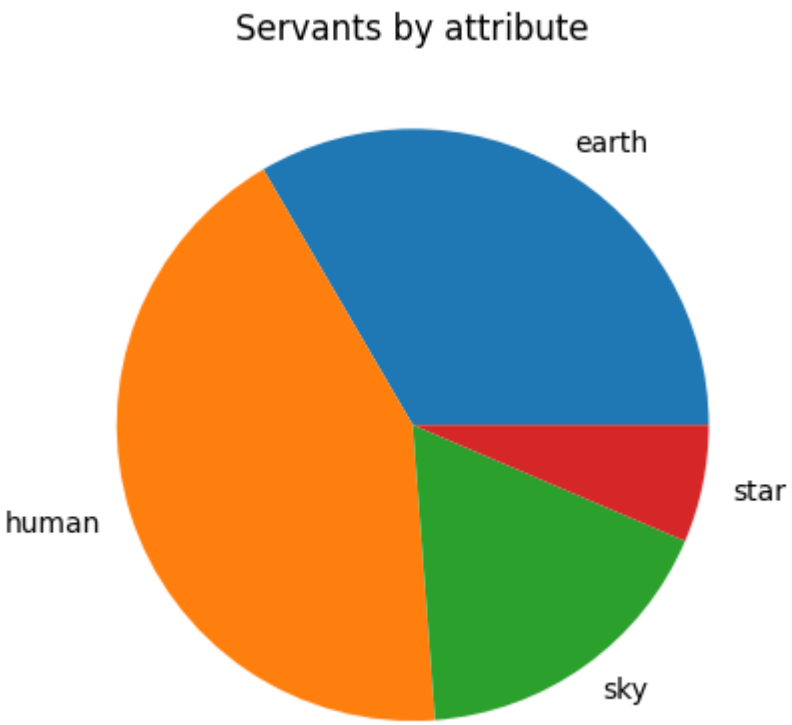
In [5]:
```
servants_per_class = df.groupby("className").count().sort_values(by="id", ascending
classes = list(servants_per_class.keys())
values = servants_per_class.values
plt.barh(classes, values)
plt.xlabel("Servant classes")
plt.ylabel("Servants number")
plt.title("Servants count by class")
plt.show()

# Griglia, colori diversi per classi extra, capire cosa fare con le mono (aggregarl
```



In [6]:
```
# Torta con extra classes vs normali vs beast?
```

Servants by attribute

In [7]:
```
servants_by_attribute = cleanData[cleanData['attribute'] != "beast"].groupby('attri
plt.pie(servants_by_attribute["id"].values,labels=servants_by_attribute["id"].keys(
plt.title("Servants by attribute")
plt.show()
```



Statistics by rarity

In [8]:
```
a = cleanData.groupby("rarity").count().filter(["id"])
a
```

Out[8]:

rarity	id
0	1
1	12
2	15
3	47
4	156
5	147

In [9]:
```
b = pd.DataFrame({"rarity":[0,1,2,3,4,5],"id":[0,0,0,0,0,0]}).set_index("rarity")
b
```

Out[9]:

rarity	id
0	0
1	0
2	0
3	0
4	0
5	0

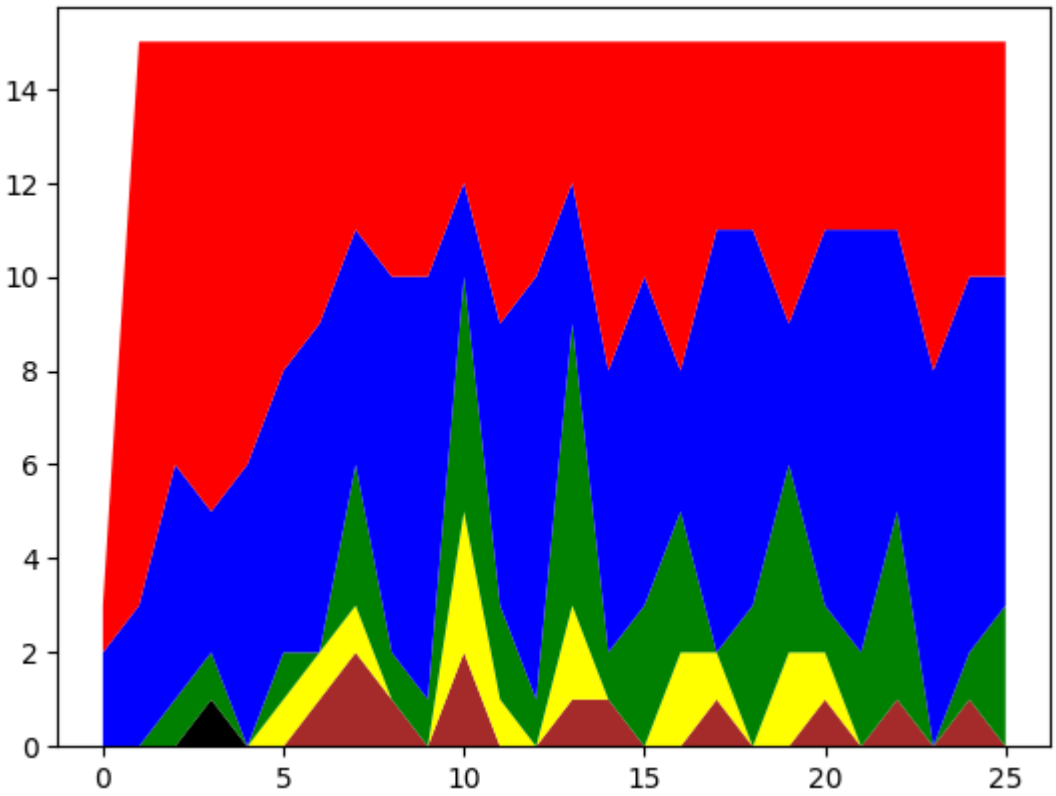
In [10]:
```
a.add(b, fill_value=0)
```

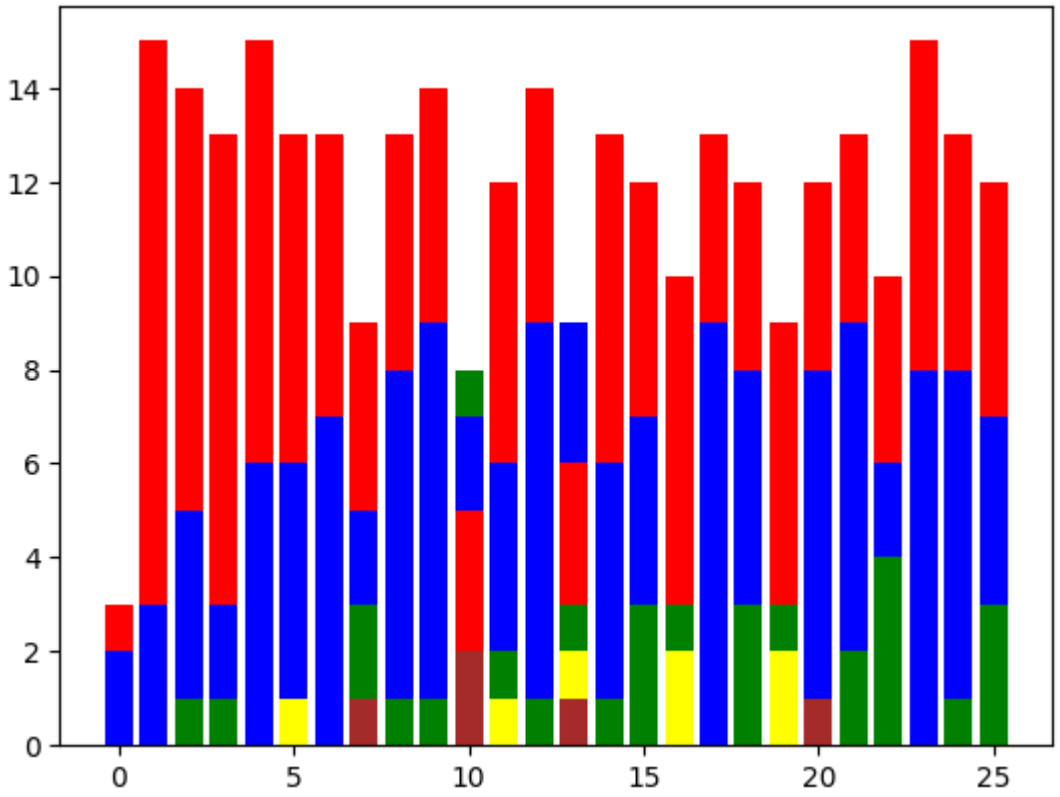
Out[10]:

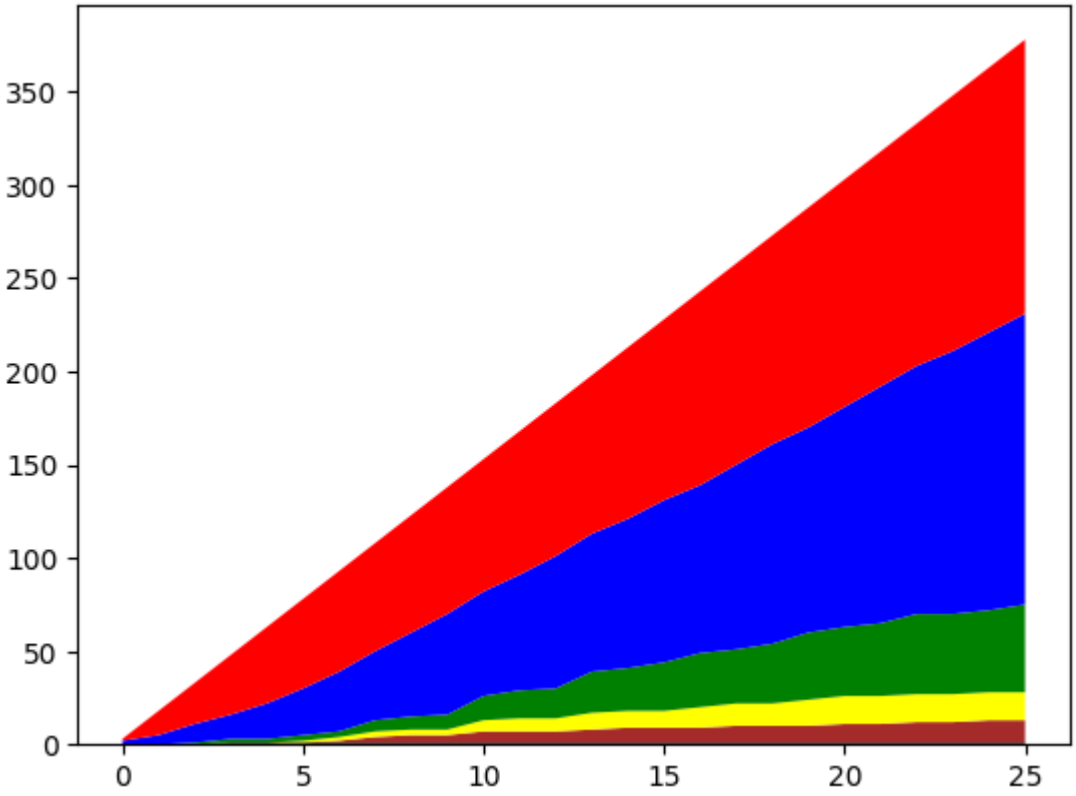
rarity	id
0	1
1	12
2	15
3	47
4	156
5	147

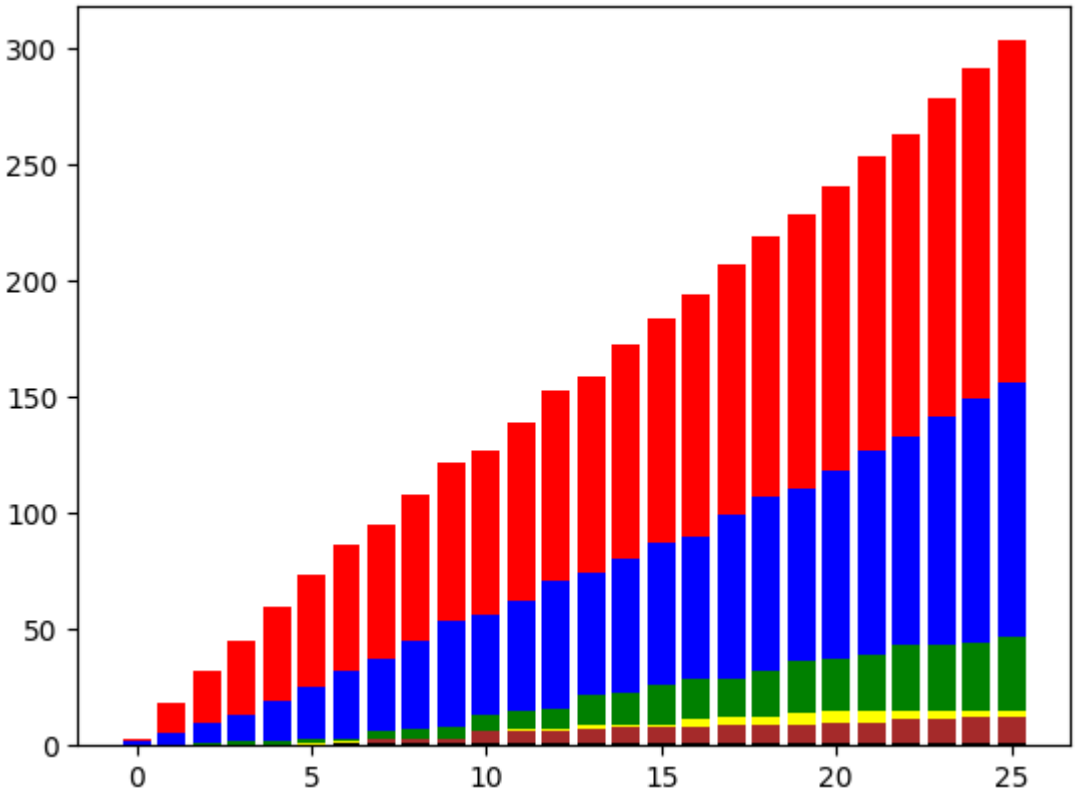
In [34]:
```
ssr_color = 'red'
sr_color = 'blue'
r_color = 'green'
b2_color = 'yellow'
b1_color = 'brown'
b0_color = 'black'

years = 8
blocks_per_year = 3

n = len(cleanData) // (blocks_per_year*years)
blocks = [cleanData[i:i+n] for i in range(0, len(cleanData),n)]

empty = pd.DataFrame({"rarity":[0,1,2,3,4,5],"id":[0,0,0,0,0,0]}).set_index("rarity

groups = [list(i.groupby("rarity").count().filter(["id"]).add(empty, fill_value=0).

groups.reverse()

x = range(0, len(blocks))
y = groups[0]
for i in groups[1:]:
    y = np.concatenate((y, i), axis=1)
```

```
plt.stackplot(x, y, labels=["0", "1", "2", "3", "4", "5"], colors=[b0_color, b1_col
plt.show()

plt.bar(x, y[0], color=b0_color)
plt.bar(x, y[1], bottom=y[0], color=b1_color)
plt.bar(x, y[2], bottom=y[1], color=b2_color)
plt.bar(x, y[3], bottom=y[2], color=r_color)
plt.bar(x, y[4], bottom=y[3], color=sr_color)
plt.bar(x, y[5], bottom=y[4], color=ssr_color)
plt.show()

y = y.cumsum(axis = 1)

plt.stackplot(x, y, labels=["0", "1", "2", "3", "4", "5"], colors=[b0_color, b1_col
plt.show()

plt.bar(x, y[0], color=b0_color)
plt.bar(x, y[1], bottom=y[0], color=b1_color)
plt.bar(x, y[2], bottom=y[1], color=b2_color)
plt.bar(x, y[3], bottom=y[2], color=r_color)
plt.bar(x, y[4], bottom=y[3], color=sr_color)
plt.bar(x, y[5], bottom=y[4], color=ssr_color)
plt.show()
```

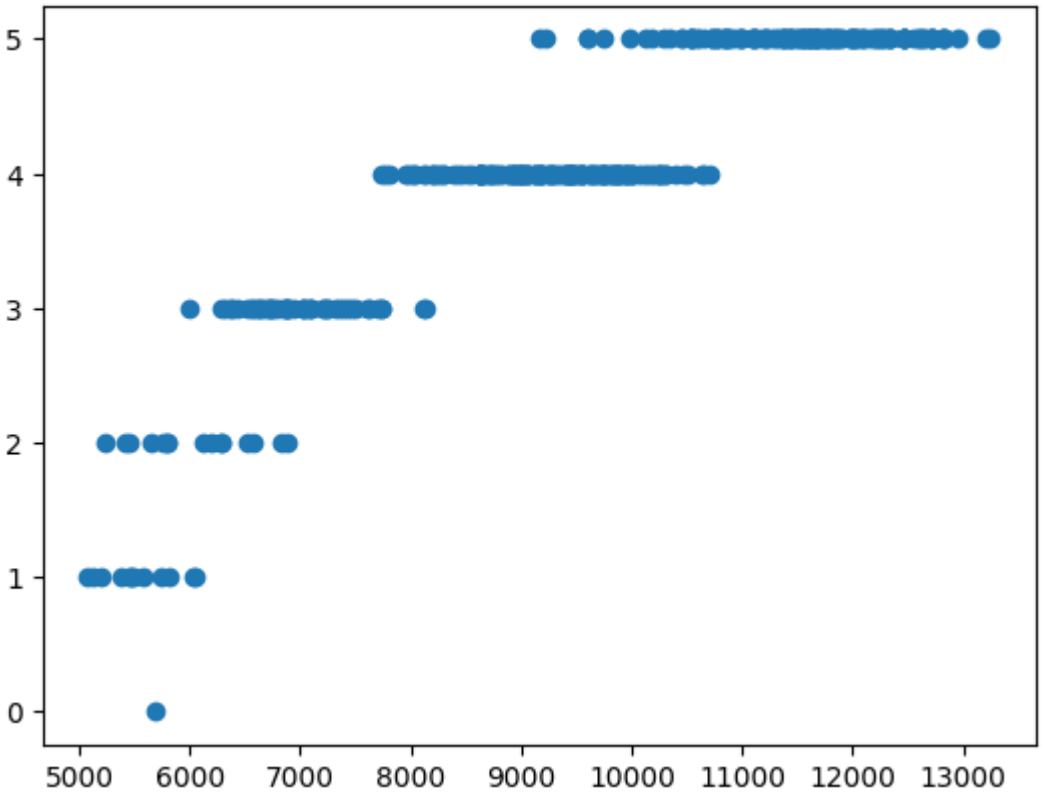


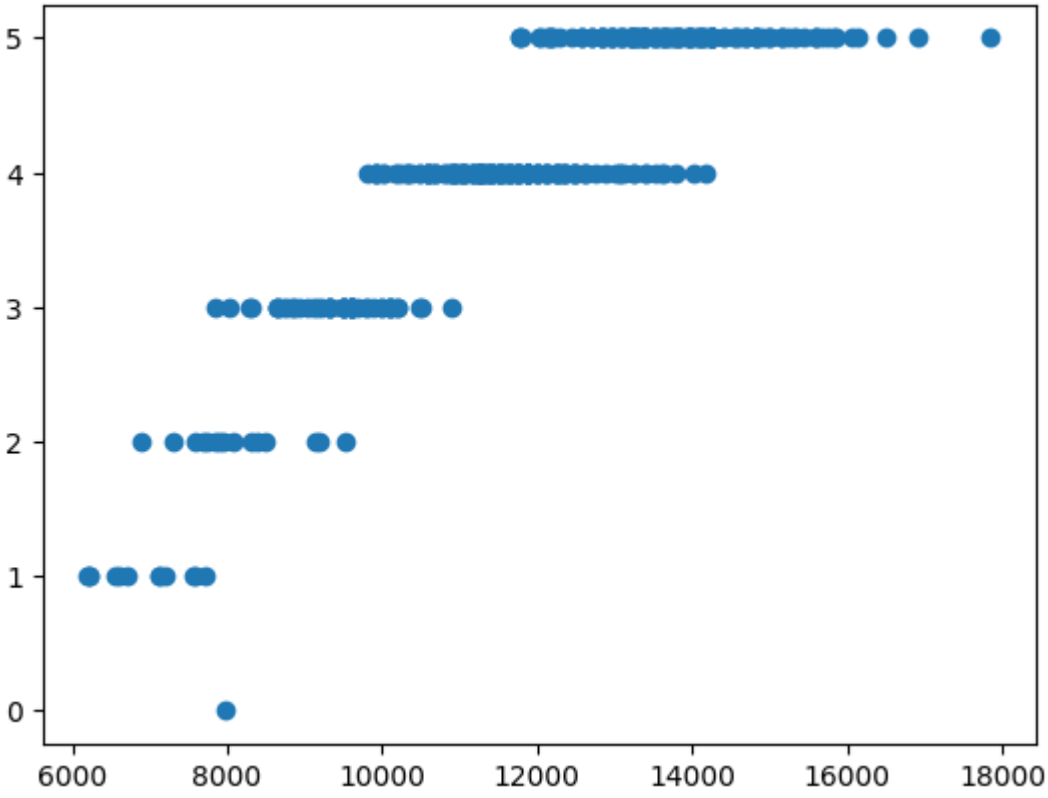

0
2
4
6
8
10
12
14
0
5
10
15
20
25

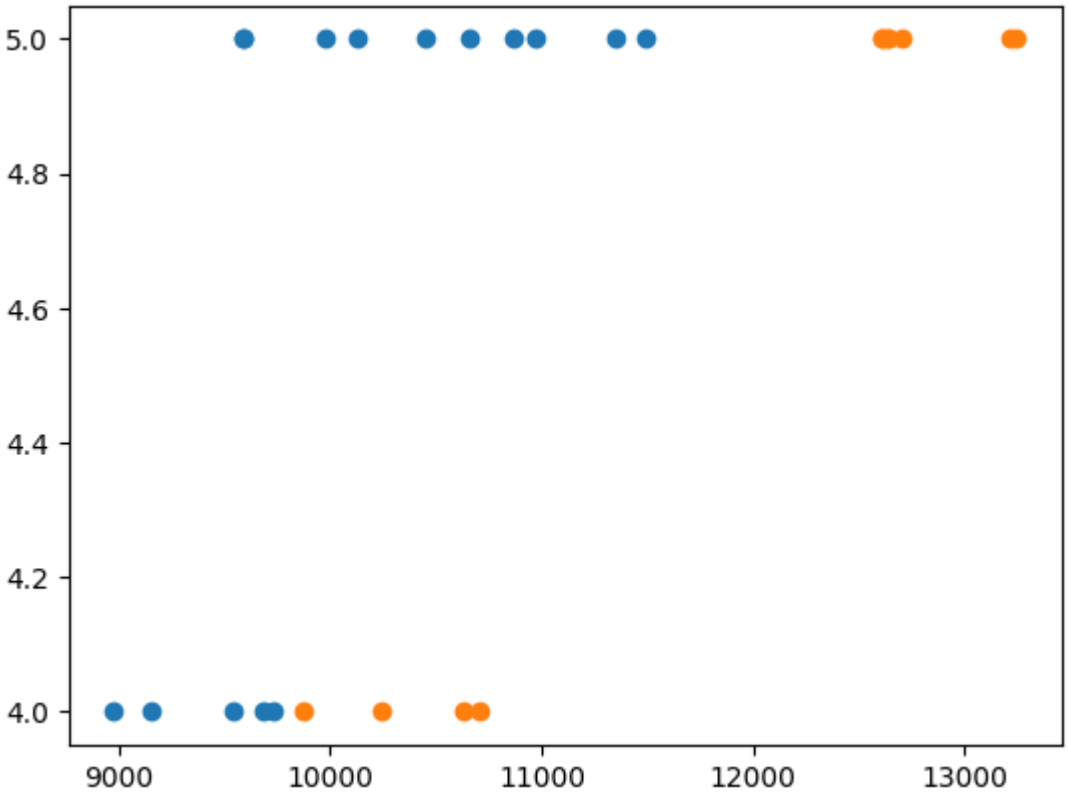

0
50
100
150
200
250
300
350
0
5
10
15
20
25



In [12]:
```
# Confronto rarita' e statistiche
# Confronto classe e statistiche
servants_statistics_clean = cleanData[(cleanData["atkMax"] < 20000) &(cleanData["hp
plt.scatter(servants_statistics_clean["atkMax"].values, servants_statistics_clean["
plt.show()

plt.scatter(servants_statistics_clean["hpMax"].values, servants_statistics_clean["r
plt.show()

# Ruler vs Avenger in hp e danni, da rivedere
servants_ruler = servants_statistics_clean[(servants_statistics_clean["className"]
servants_avenger = servants_statistics_clean[(servants_statistics_clean["className"
plt.scatter(servants_ruler["atkMax"].values, servants_ruler["rarity"].values)
plt.scatter(servants_avenger["atkMax"].values, servants_avenger["rarity"].values)
plt.show()
```


5
4
3
2
1
0
5000
6000
7000
8000
9000
10000
11000
12000
13000


5
4
3
2
1
0
6000
8000
10000
12000
14000
16000
18000

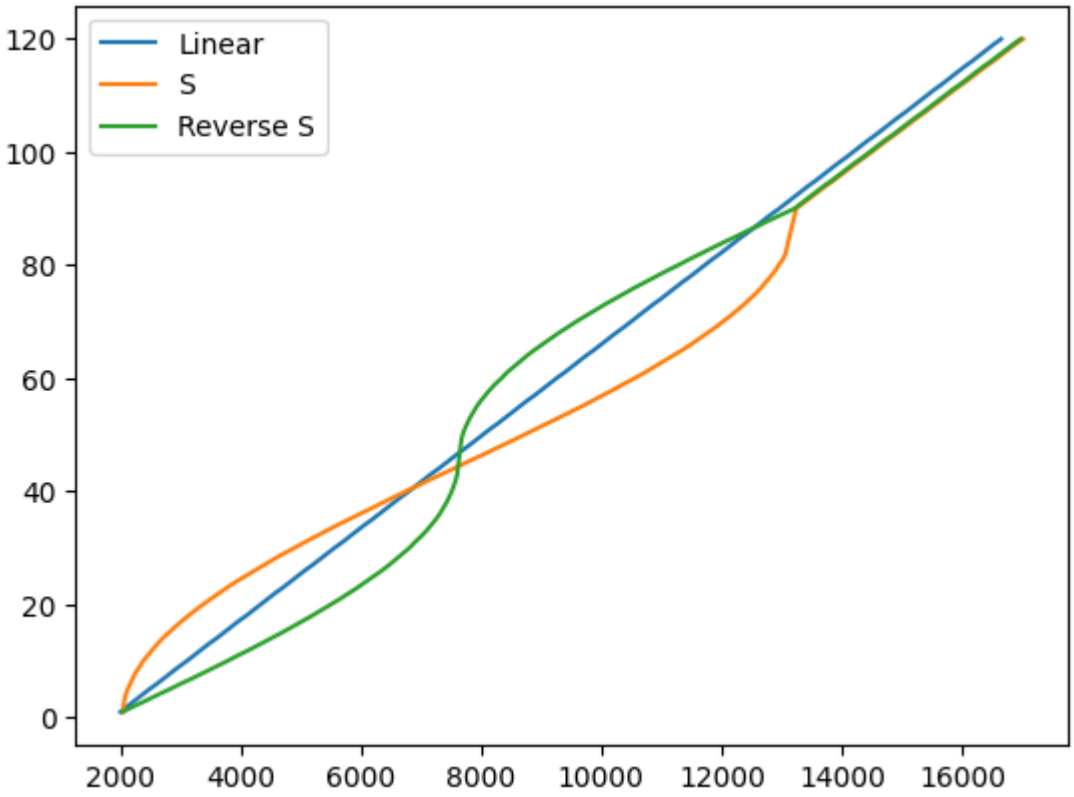


In [13]:
```
# NP type (danno o meno) e correlazione con l'essere caster
```

Statistics Growth

In [14]:
```
linear_growth = cleanData.loc[(cleanData['name'] == "Ibuki-Douji") & (cleanData['cl
s_growth = cleanData.loc[(cleanData['name'] == "Jeanne d'Arc (Alter)") & (cleanData
reverse_s_growth = cleanData.loc[(cleanData['name'] == "Kama") & (cleanData['classN
print(len(reverse_s_growth))

plt.plot(linear_growth, list(range(1, len(linear_growth)+1)))
plt.plot(s_growth, list(range(1, len(s_growth)+1)))
plt.plot(reverse_s_growth, list(range(1, len(reverse_s_growth)+1)))
plt.legend(["Linear", "S", "Reverse S"])
plt.show()
```

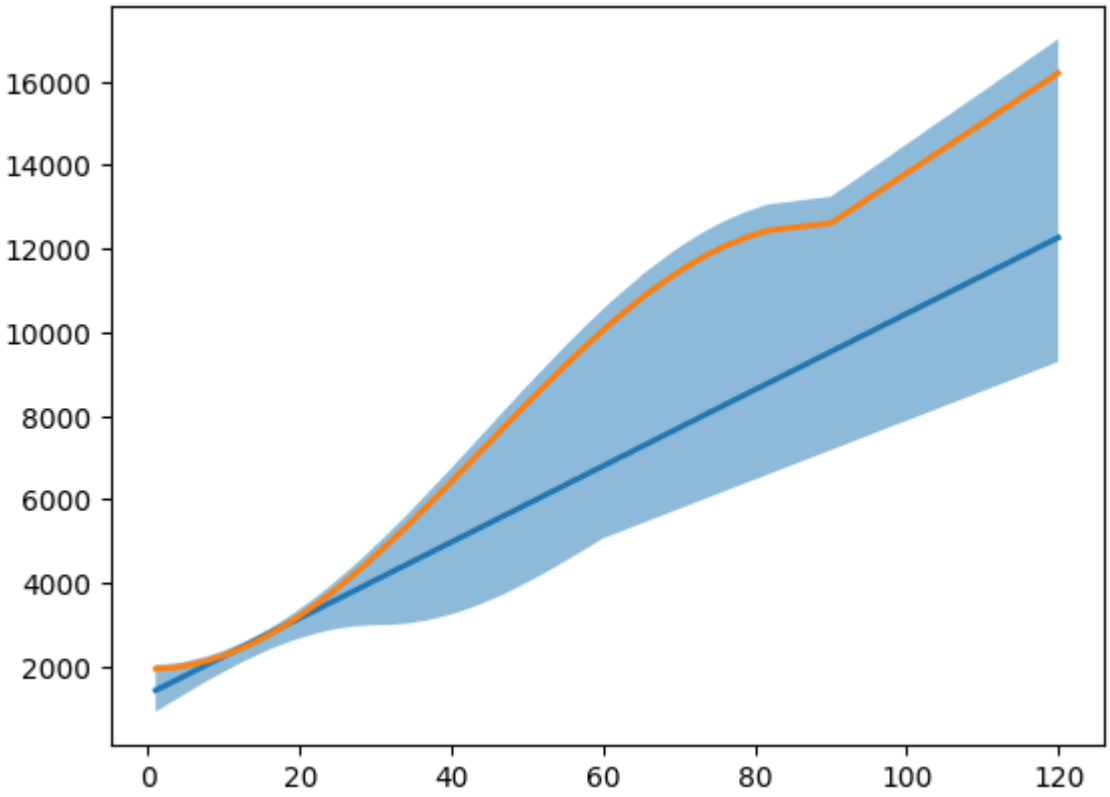
```
120
```



Classes with the best and worst statistics

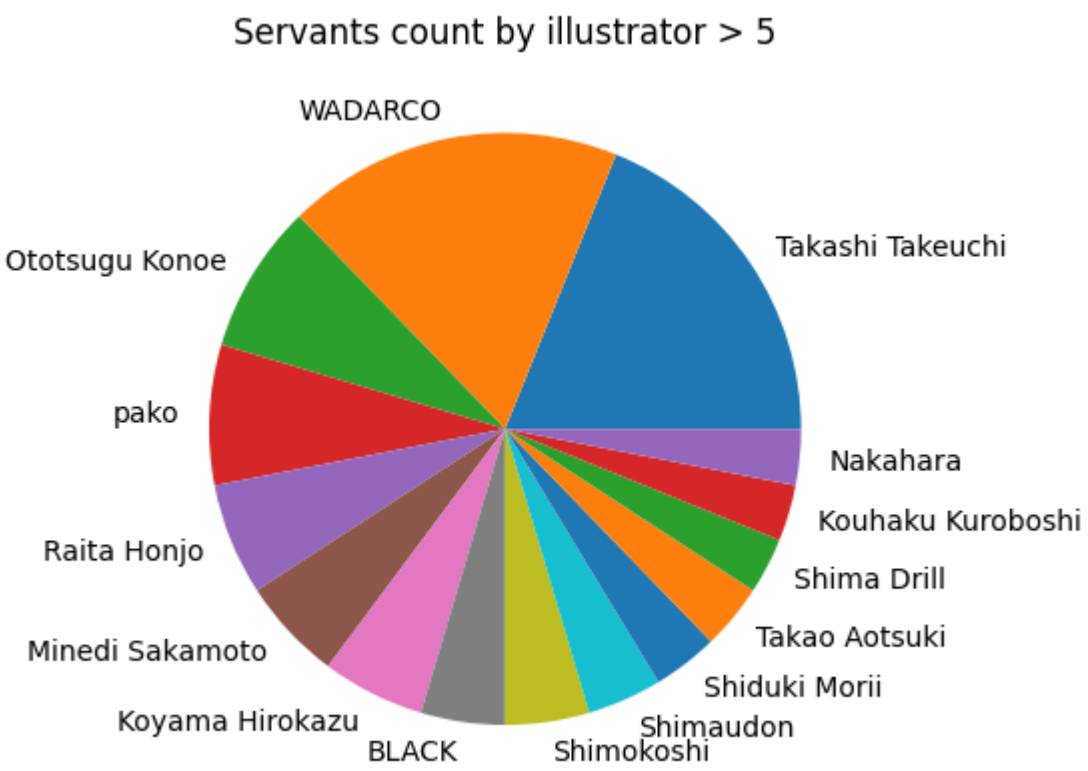
In [15]:

```
max_atk_growth = cleanData[cleanData["atkMax"] == cleanData["atkMax"].max()].filter
min_atk_growth = cleanData[cleanData["atkMax"] == cleanData["atkMax"].min()].filter

casters = cleanData[cleanData["className"] == "caster"]
caster_average_growth = casters[casters["atkMax"] == casters["atkMax"].median()].il
avengers = cleanData[cleanData["className"] == "avenger"]
avenger_average_growth = avengers[avengers["atkMax"] == avengers["atkMax"].median()


to_120 = range(1, 120+1)

fig, ax = plt.subplots()
ax.fill_between(to_120, min_atk_growth, max_atk_growth, alpha=.5, linewidth=0)
ax.plot(to_120, caster_average_growth, linewidth=2)
ax.plot(to_120, avenger_average_growth, linewidth=2)

plt.show()
```



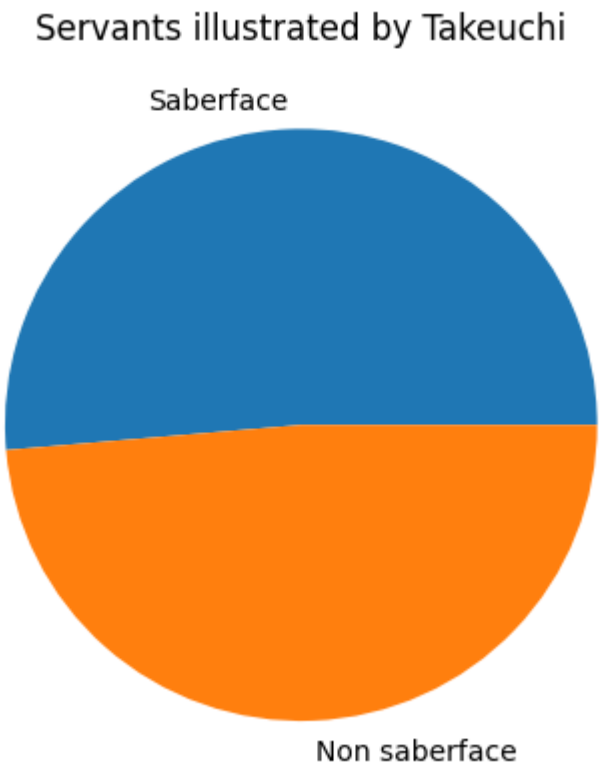
Servants by illustrator

In [16]:
```
servants_by_illustrator = cleanData.groupby(by='illustrator').count().sort_values(b
servants_by_illustrator  = servants_by_illustrator[servants_by_illustrator > 5]
illustrators = list(servants_by_illustrator.keys())
values = servants_by_illustrator.values
plt.pie(values,labels=illustrators)
plt.title("Servants count by illustrator > 5")
plt.show()
```



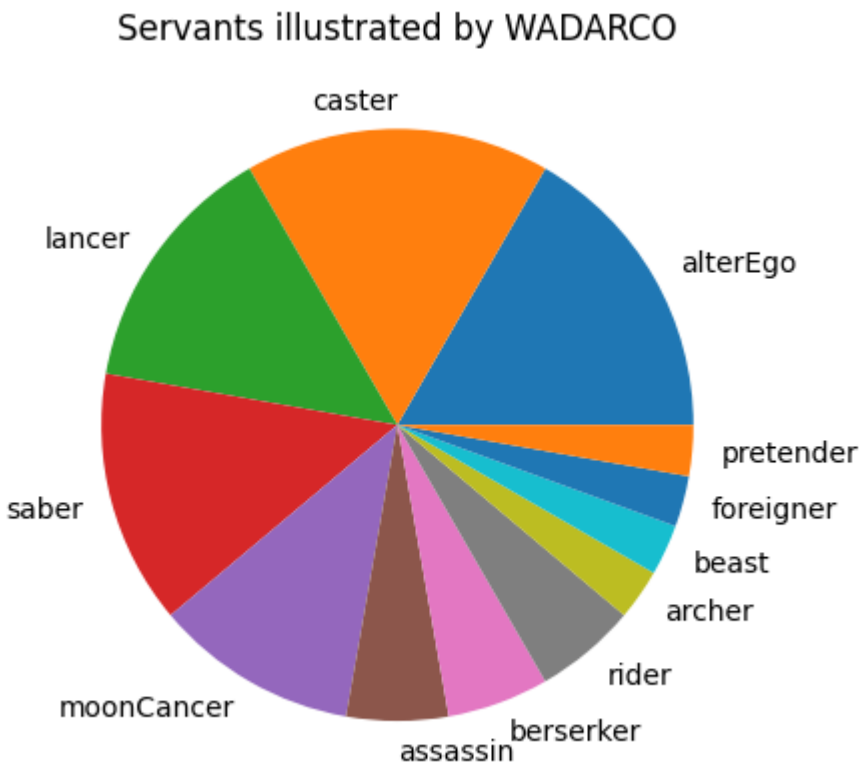
Servants illustrated by Takeuchi by the presence of the "saberface" trait

In [17]:
```
servants_by_take = cleanData[cleanData["illustrator"] == "Takashi Takeuchi"].copy()
servants_by_take = servants_by_take.groupby(by='saberface').count().sort_values(by=
values = servants_by_take["id"].values
plt.pie(values,labels=["Saberface", "Non saberface"])
plt.title("Servants illustrated by Takeuchi")
plt.show() #renderlo a barre e far vedere la distribuzione totale vs quella di take
```



Servants illustrated by WADARCO by class being extra or not

In [18]:
```
servants_by_wada = cleanData[cleanData["illustrator"] == "WADARCO"].copy()
servants_by_wada = servants_by_wada.groupby(by='className').count().sort_values(by=
values = servants_by_wada["id"].values
plt.pie(values,labels=servants_by_wada["id"].keys())
plt.title("Servants illustrated by WADARCO")
plt.show()
```


Servants illustrated by WADARCO
caster
alterEgo
lancer
pretender
foreigner
saber
beast
archer
rider
moonCancer
berserker
assassin

In [ ]: