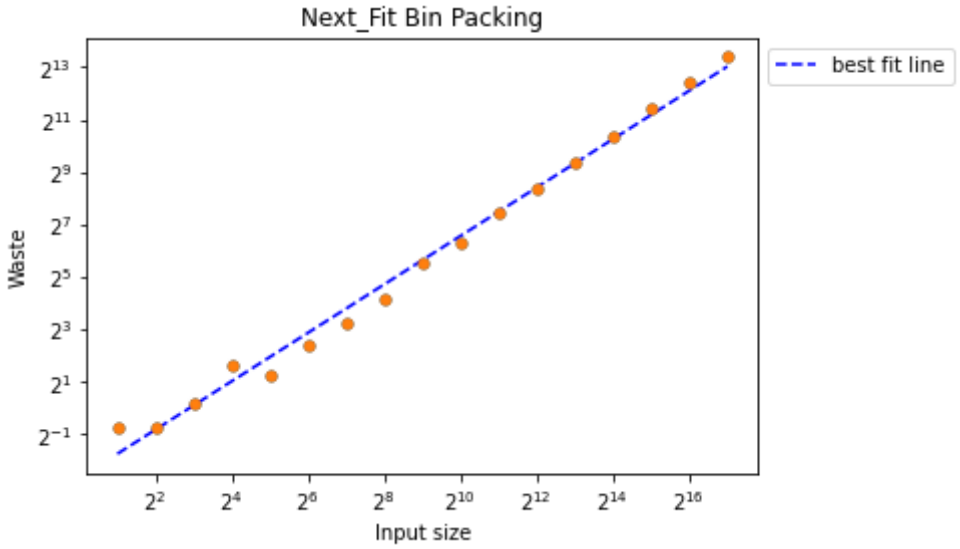
In [106…
```
import pandas as pd
import numpy as np
from math import log, e
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
```

In [107…
```
df_nf = pd.read_csv("next_fit.csv",header=None, names=["Size","Waste"])
df_ff = pd.read_csv("first_fit.csv",header=None, names=["Size","Waste"])
df_ffd = pd.read_csv("first_fit_dec.csv",header=None, names=["Size","Waste"])
df_bf = pd.read_csv("best_fit.csv",header=None, names=["Size","Waste"])
df_bfd = pd.read_csv("best_fit_dec.csv",header=None, names=["Size","Waste"])
```

In [108…
```
df_nf.plot(x="Size", y="Waste", kind = "scatter")

x = df_nf['Size']
y = df_nf['Waste']

logx = np.log(x)
logy = np.log(y)

m, b = np.polyfit(logx, logy, 1)
print("best fit line slope: ", m)
fit = np.poly1d((m, b))
expected_logy = fit(logx)

p = plt.loglog(x, y, '.', base = 2, markersize = 10)

r2 = r2_score(logy, expected_logy)

fit_p = plt.loglog(x[::len(x)-1], (e ** expected_logy)[::len(y)-1], '--', base =
label = "best fit line",
markersize = 6, color = "blue")

x2 = df_nf['Size']
y2 = df_nf['Waste']
p_random = plt.loglog(x2, y2, '.', base = 2, markersize = 10)
plt.title("Next_Fit Bin Packing")
plt.xlabel('Input size')
plt.ylabel('Waste')
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')
```

```
best fit line slope:  0.924518781228405
```

Out[108…
```
<matplotlib.legend.Legend at 0x7febabab9460>
```



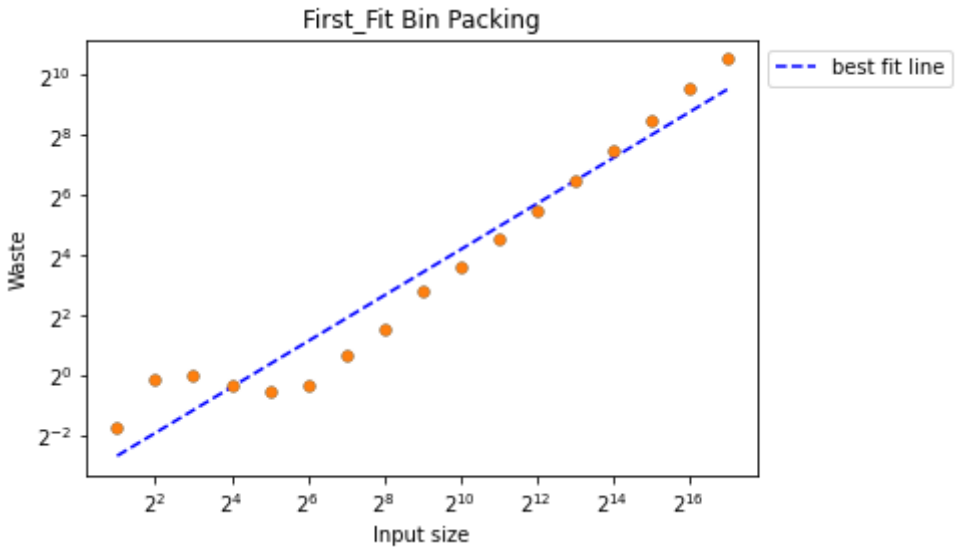
In [109…

```
df_ff.plot(x="Size", y="Waste", kind = "scatter")

x = df_ff['Size']
y = df_ff['Waste']

logx = np.log(x)
logy = np.log(y)

m, b = np.polyfit(logx, logy, 1)
print("best fit line slope: ", m)
fit = np.poly1d((m, b))
expected_logy = fit(logx)

p = plt.loglog(x, y, '.', base = 2, markersize = 10)

r2 = r2_score(logy, expected_logy)

fit_p = plt.loglog(x[::len(x)-1], (e ** expected_logy)[::len(y)-1], '--', base =
label = "best fit line",
markersize = 6, color = "blue")

x2 = df_ff['Size']
y2 = df_ff['Waste']
p_random = plt.loglog(x2, y2, '.', base = 2, markersize = 10)
plt.title("First_Fit Bin Packing")
plt.xlabel('Input size')
plt.ylabel('Waste')
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')
```

```
best fit line slope:  0.7612513392249503
```

Out[109…

```
<matplotlib.legend.Legend at 0x7febabc17a00>
```



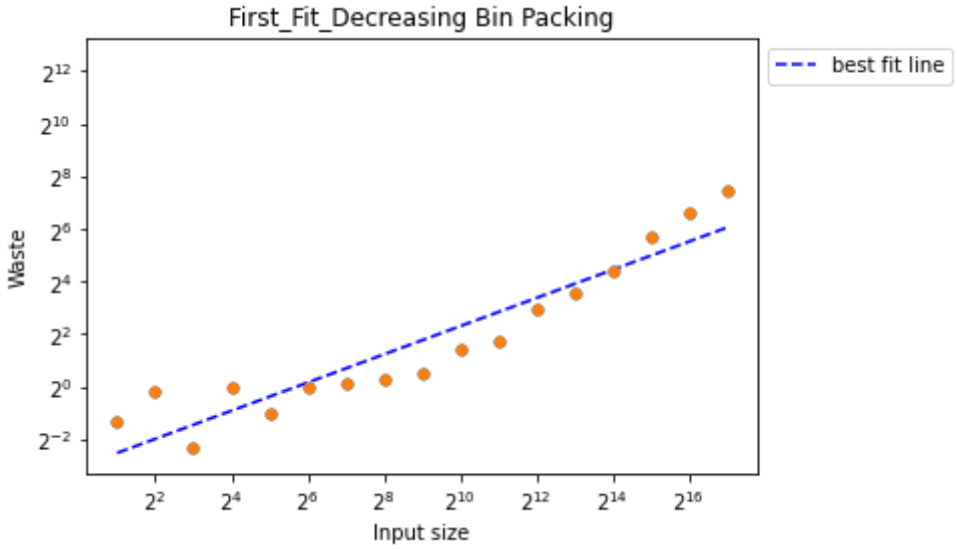
In [110…

```
df_ffd.plot(x="Size", y="Waste", kind = "scatter")

x = df_ffd['Size']
y = df_ffd['Waste']

logx = np.log(x)
logy = np.log(y)

m, b = np.polyfit(logx, logy, 1)
print("best fit line slope: ", m)
fit = np.poly1d((m, b))
expected_logy = fit(logx)

p = plt.loglog(x, y, '.', base = 2, markersize = 10)

r2 = r2_score(logy, expected_logy)

fit_p = plt.loglog(x[::len(x)-1], (e ** expected_logy)[::len(y)-1], '--', base =
label = "best fit line",
markersize = 6, color = "blue")

x2 = df_ffd['Size']
y2 = df_ffd['Waste']
p_random = plt.loglog(x2, y2, '.', base = 2, markersize = 10)
plt.title("First_Fit_Decreasing Bin Packing")
plt.xlabel('Input size')
plt.ylabel('Waste')
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')
ax = plt.gca()
ax.set_ylim([0.1, 10000])
```

```
best fit line slope:  0.5364382025225617
```

Out[110…

```
(0.1, 10000)
```



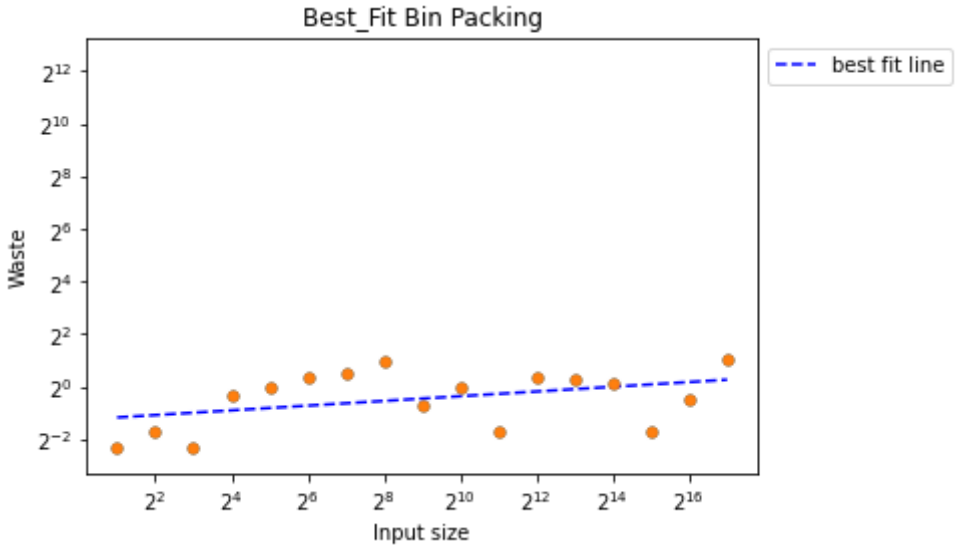
In [111…

```
df_bf.plot(x="Size", y="Waste", kind = "scatter")

x = df_bf['Size']
y = df_bf['Waste']

logx = np.log(x)
logy = np.log(y)

m, b = np.polyfit(logx, logy, 1)
print("best fit line slope: ", m)
fit = np.poly1d((m, b))
expected_logy = fit(logx)

p = plt.loglog(x, y, '.', base = 2, markersize = 10)

r2 = r2_score(logy, expected_logy)

fit_p = plt.loglog(x[::len(x)-1], (e ** expected_logy)[::len(y)-1], '--', base =
label = "best fit line",
markersize = 6, color = "blue")

x2 = df_bf['Size']
y2 = df_bf['Waste']
p_random = plt.loglog(x2, y2, '.', base = 2, markersize = 10)
plt.title("Best_Fit Bin Packing")
plt.xlabel('Input size')
plt.ylabel('Waste')
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')
ax = plt.gca()
ax.set_ylim([0.1, 10000])
```

```
best fit line slope:  0.0897560249217488
```

Out[111…

```
(0.1, 10000)
```



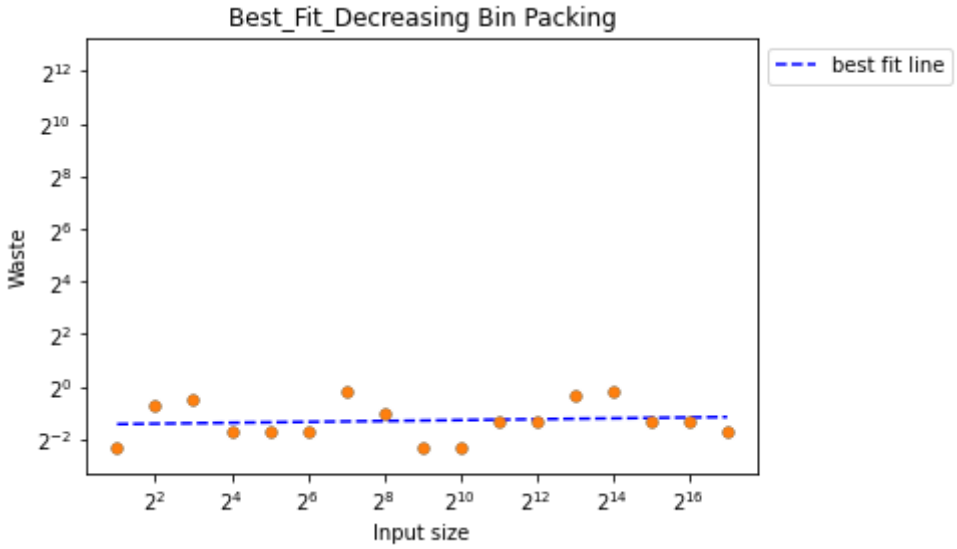
In [112…

```
df_bfd.plot(x="Size", y="Waste", kind = "scatter")

x = df_bfd['Size']
y = df_bfd['Waste']

logx = np.log(x)
logy = np.log(y)

m, b = np.polyfit(logx, logy, 1)
print("best fit line slope: ", m)
fit = np.poly1d((m, b))
expected_logy = fit(logx)
p = plt.loglog(x, y, '.', base = 2, markersize = 10)

r2 = r2_score(logy, expected_logy)

fit_p = plt.loglog(x[::len(x)-1], (e ** expected_logy)[::len(y)-1], '--', base =
label = "best fit line",
markersize = 6, color = "blue")

x2 = df_bfd['Size']
y2 = df_bfd['Waste']
p_random = plt.loglog(x2, y2, '.', base = 2, markersize = 10)
plt.title("Best_Fit_Decreasing Bin Packing")
plt.xlabel('Input size')
plt.ylabel('Waste')
plt.legend(bbox_to_anchor=(1, 1), loc='upper left')
ax = plt.gca()
ax.set_ylim([0.1, 10000])
```

```
best fit line slope:  0.016934141836179483
```

Out[112…

```
(0.1, 10000)
```



Without a doubt, I found the best algorithm to be best fit decreasing. It had significantly less waste than the other algorithms with most waste under 1. It also had by far the lowest slope amongst the algorithms (printed above each graph below the code). This does not surprise me because best fit seems to be the optimal way of approaching this situation. Putting the vector in decreasing order before running the algorithm makes it even better because the bigger items are put in first. This helps because as you start going down the list, smaller items start showing up and it is easier to find a spot for them because 0.1 and 0.2 don't require much space. Towards the end ideally, most bins are almost full. If that's the case, getting a big item such as 0.6 would force a new bin to be created. This is why decreasing order makes such a big difference. Best fit has the best approach because it decreases the amount of potential waste. Best fit can beat first fit in situations where the items are [0.6, 0.7, 0.3, 0.4]. In first fit, this would need 3 bins but best fit would only need 2 bins. Next fit was the clear worst algorithm in this experiment. This was expected because as a new bin is created, all existing beens are ignored which can cause a lot of waste. For example, [0.5, 0.6, 0.4, 0.1] although the first bin still has 0.5 space open, it can never be accessed because the item after it caused an overflow.