Input Analysis

1) We first begin by noticing that we have 10,000 data points
2) Furthermore we check for any NA values in the columns

```
df.isna().sum()
```

```
Arrival no             0
Arrival time (sec)     0
Base station           0
Call duration (sec)    0
velocity (km/h)        0
dtype: int64
```

3) Checking for any anomalies in the features

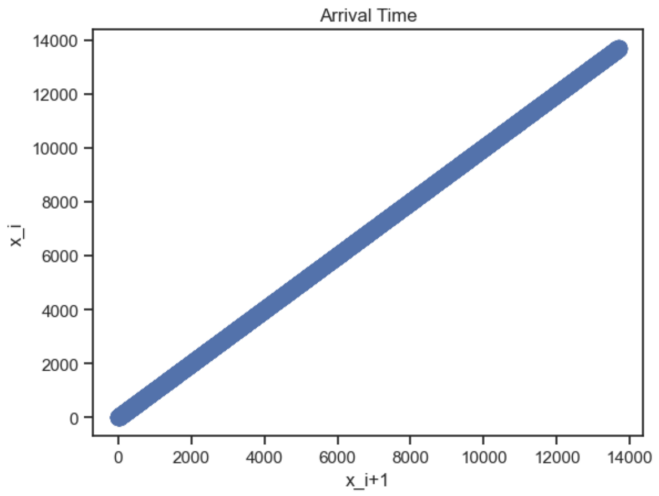
```
In [5]: df.describe()
```

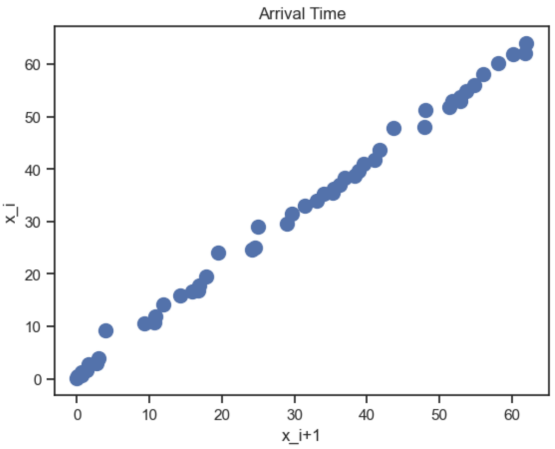
Out[5]:

	Arrival no	Arrival time (sec)	Base station	Call duration (sec)	velocity (km/h)
count	10000.00000	10000.000000	10000.00000	10000.000000	10000.000000
mean	5000.50000	6838.187985	10.54220	109.835901	120.072098
std	2886.89568	3955.015689	5.78662	99.138916	9.019058
min	1.00000	0.000000	1.00000	10.003952	86.423152
25%	2500.75000	3401.852699	6.00000	38.777914	113.892676
50%	5000.50000	6803.665666	11.00000	79.937560	120.021618
75%	7500.25000	10257.006519	16.00000	148.109589	126.188008
max	10000.00000	13696.799448	20.00000	1239.836192	151.501467

4) We notice that Call Duration might be left skewed since the mean is a lot bigger than median (109.83 vs 79.94)
5) We can now assess independence of the features by creating scatterplots of the pairs (X_i, X_{i+1})

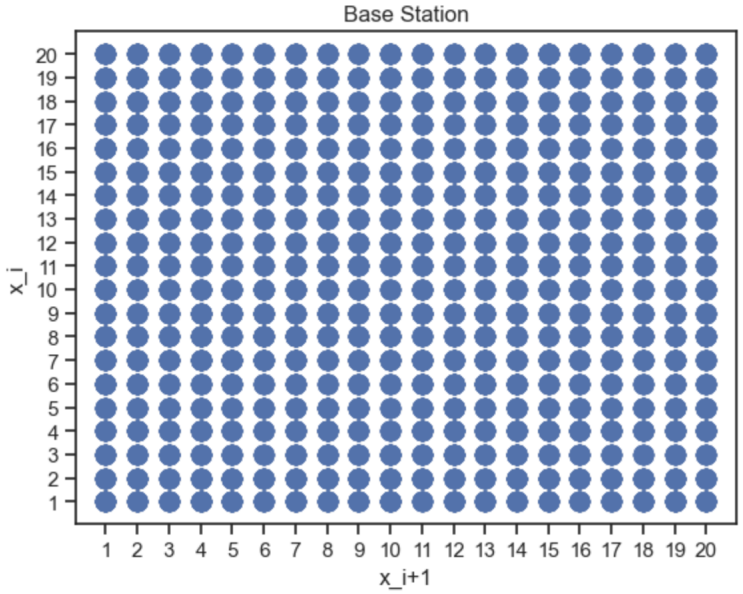
```
In [29]: #Assessing Independence
plt.title('Arrival Time')
plt.xlabel('x_i+1')
plt.ylabel('x_i')
plt.scatter(x= df.iloc[:-1,1:2], y= df.iloc[1:,1:2], s = 100)
plt.show()
```

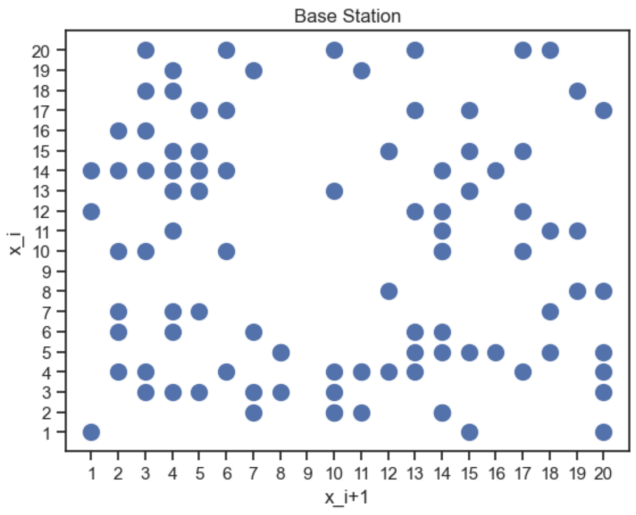




a) Arrival Time is positively correlated which is because x_i+1 are always >= x_i, since time moves forward linearly. Furthermore, we notice the slight discrepancies in the correlation when we plot for a smaller range

```
In [45]: #Assessing Independence
         plt.title('Base Station')
         plt.xticks(ticks=[1,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
         plt.yticks(ticks=[1,2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
         plt.xlabel('x_i+1')
         plt.ylabel('x_i')
         plt.scatter(x= df.iloc[:-1,2:3], y= df.iloc[1:,2:3], s = 100)
         plt.show()
```

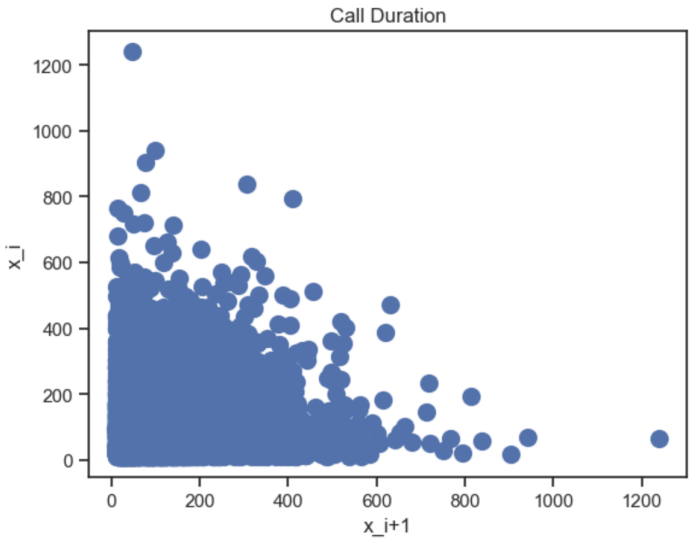




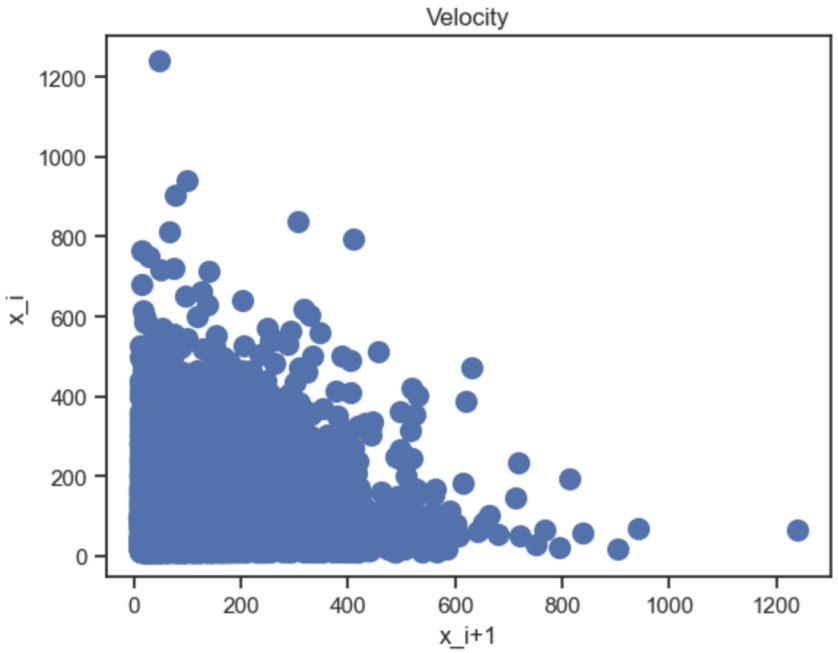
of values, which tells us the inter-arrival times of the cars is random and not uniform.

b) For Base Station we can see from the second scatterplot which only has 100 values plotted that the points are scattered and independent of each other.

```
In [48]: #Assessing Independence
         plt.title('Call Duration')
         plt.xlabel('x_i+1')
         plt.ylabel('x_i')
         plt.scatter(x= df.iloc[:-1,3:4], y= df.iloc[1:,3:4], s = 100)
         plt.show()
```



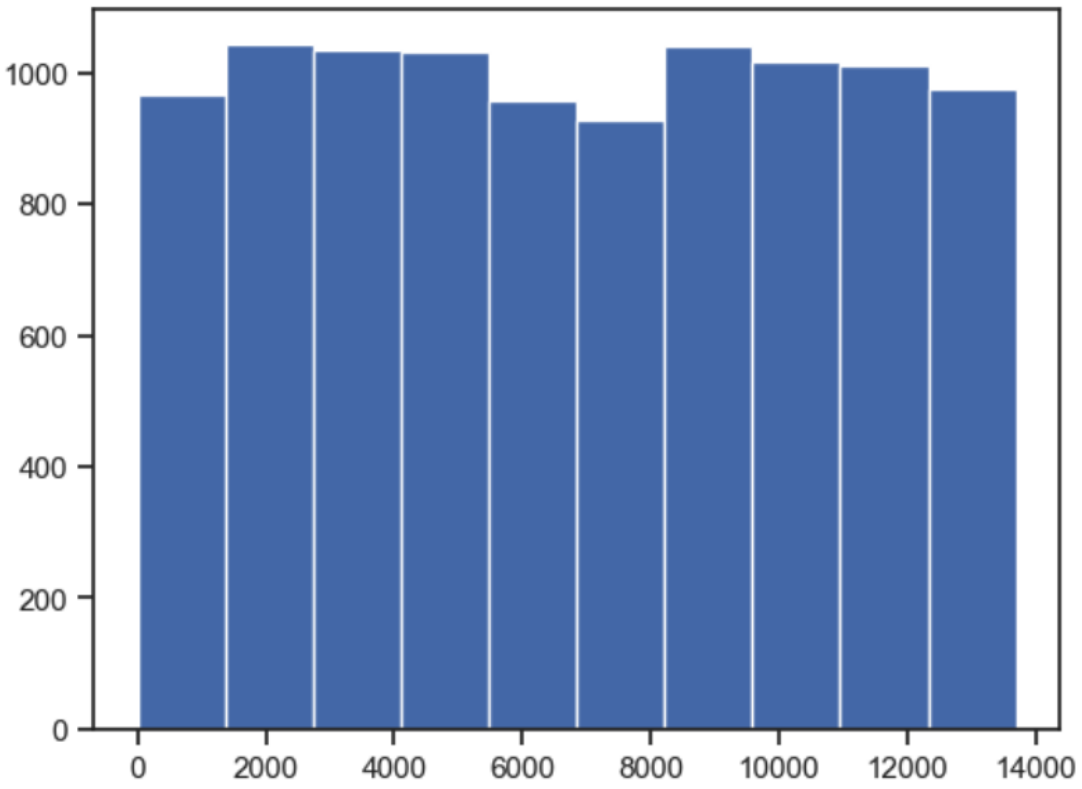
c) For Call Duration we can see from the points that they are all randomly scattered and independent of each other



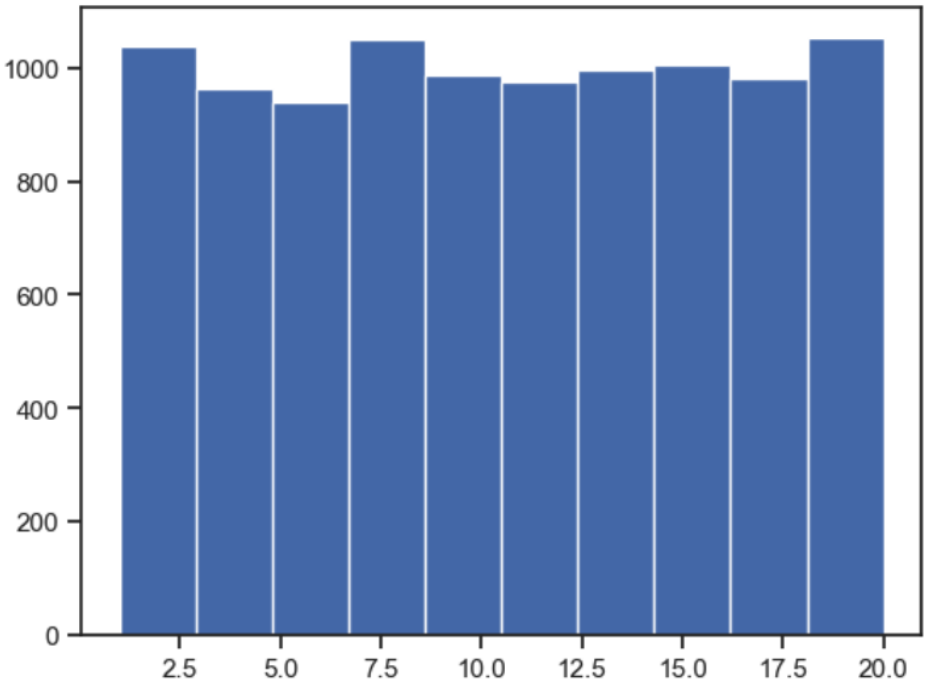
d) For Velocity we can see the plot is like Call Duration, and the points are randomly scattered and independent of each other.

6) Plot Histogram of the features: Arrival time, Base Station, Call duration, Velocity

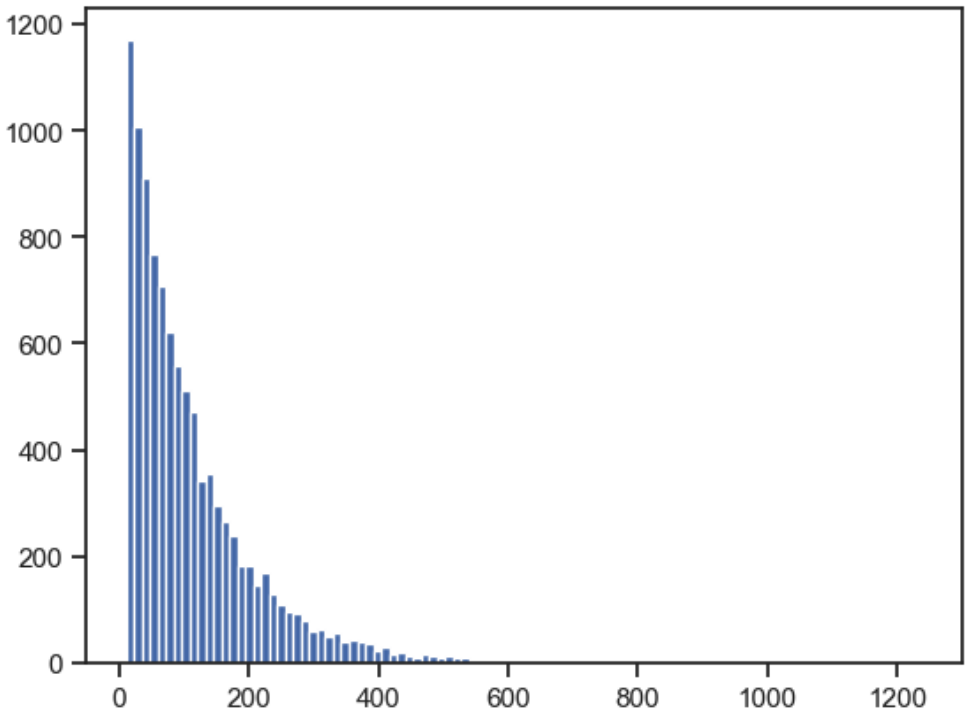
```
plt.hist(df['Arrival time (sec)'])
plt.show()
```



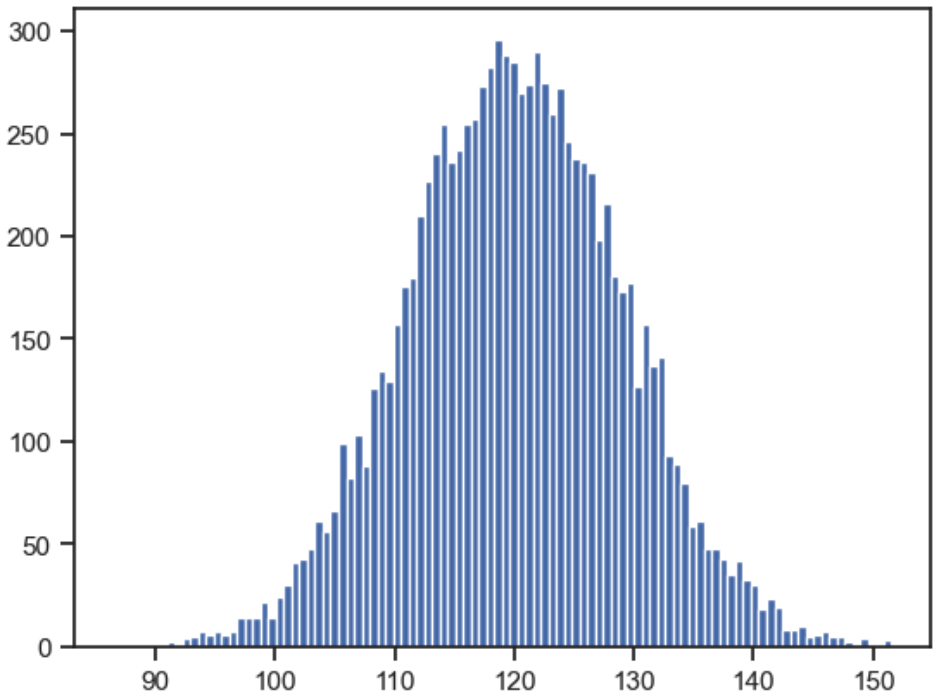
In [135]:
```
plt.hist(df['Base station '])
plt.show()
```



```
In [138]: plt.hist(df['Call duration (sec)'], bins=100)
          plt.show()
```




```
In [139]: plt.hist(df['velocity (km/h)'], bins=100)
          plt.show()
```



a) Looking at the histograms of each of the features we approximate the distributions to be the following:
   i) For Arrival Time we can see that the distribution is uniformly distributed

ii) For Base Station we can see that the distribution is uniformly distributed
iii) For Call Duration we can see that the distribution is exponentially distributed
iv) For Velocity we can see that the distribution looks close to a normal distribution

7) Estimate the parameters for the distributions above

```
In [5]: df.describe()
```

Out[5]:

	Arrival no	Arrival time (sec)	Base station	Call duration (sec)	velocity (km/h)
count	10000.00000	10000.000000	10000.00000	10000.000000	10000.000000
mean	5000.50000	6838.187985	10.54220	109.835901	120.072098
std	2886.89568	3955.015689	5.78662	99.138916	9.019058
min	1.00000	0.000000	1.00000	10.003952	86.423152
25%	2500.75000	3401.852699	6.00000	38.777914	113.892676
50%	5000.50000	6803.665666	11.00000	79.937560	120.021618
75%	7500.25000	10257.006519	16.00000	148.109589	126.188008
max	10000.00000	13696.799448	20.00000	1239.836192	151.501467

a) To get the values of Max, Min, Sample Mean & Sample Variance we use the table as shown above.
b) For Arrival Time, Uniform Distribution: $(x_2-x_1)/(b-a)$. We estimate b as $\max(x_1, x_2,...,x_n)$, estimate a as $\min(x_1, x_2,...,x_n)$. Therefore b = 13696.8 and a = 0.0. This can be seen from:
c) For Base Station, Uniform Distribution, we follow the steps of Arrival Time, therefore we estimate b = 20 and a = 1. This can be seen from:
d) For Call Duration, Exponential Distribution, we estimate the parameter β by using the sample mean. Therefore β = 109.84.
e) For Velocity, Normal Distribution, we estimate mu with sample mean and standard deviation with the sample variance. Therefore mu = 120.07 and standard deviation = 9.02.

8) We now apply goodness of fit tests to each of the models we have approximated to see how they perform. We use scipy's stats module to implement the chisq test and the Kolmogorov-Smirnov test. We apply chisquare tests on Categorical Data while Kolmogorov-Smirnov test for Continuous data. The kstest provided by the library scipy in python is able to verify if two distributions are from the same distribution using the Kolmogorov-Smirnov test, as long as the data is continuous. The significance level of test we use is 0.05.
   a) For the Arrival time, we model a Unifiorm distribution with the a and b values we approximated above and then apply the Kolmogorov-Smirov test as shown below and get a pvalue of 0.66 which is greater than our significance level of 0.05 and therefore we do not reject the null

hypothesis. Therefore the Uniform distribution with a = 0.0. and b = 13696.8 is a good fit for our model.

```
In [102]: import scipy.stats as stats

          uniformDist = np.random.uniform(0.0, 13696.8, 10000)
          obs = df['Arrival time (sec)'].values.tolist()
          stats.kstest(obs, uniformDist)

Out[102]: KstestResult(statistic=0.0103, pvalue=0.6637453476707832)
```

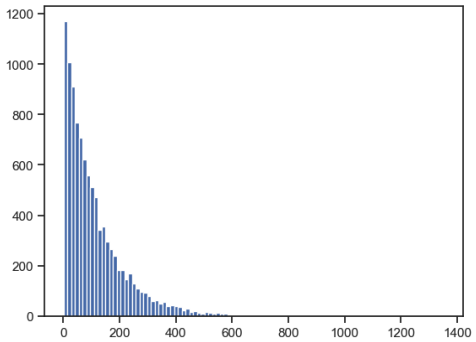
b) For Base Station, we do a similar test with a as 1 and b as 20. We get a pvalue of 0.14 which is greater than 0.05 and therefore we do not reject our null hypothesis and we can model our distribution using uniform distribution.

```
In [179]: import scipy.stats as stats
          from collections import Counter

          observed_freq = list(Counter(df['Base station ']).values())
          expected_freq = [500] * 20
          stats.chisquare(f_exp=expected_freq, f_obs=observed_freq)

Out[179]: Power_divergenceResult(statistic=25.656, pvalue=0.1400629076546357)
```

c) For Call Duration, we notice that when applying the Kolmogorov-Smirov test we get a p-value < 0.05. After further check, we can notice that there are very few values in the first bar, such that it looks non-existent. Referring to the bar in the range of 0 to the very first visible bar. Therefore we shift the values to the left by subtracting all the Call Durations with the minimum Call Durations. However, we still obtain a p-value (1.433e-08) < 0.05, as shown below. Therefore we multiply the values by 1.1, as a linear transformation, to shift the distribution to the right. histogram we can see a clear similarity in the two distributions and we are able move forward with using a exponential distribution with β = 109.84 and the values shifted to the right by 1.1. This is because the pvalue of 0.717 > 0.05.

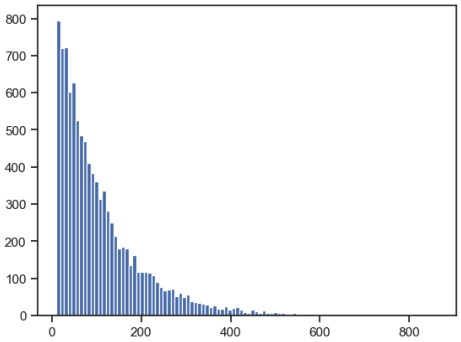
In [10]:

```
df['Call duration (sec)']
#for i, x in enumerate(df['Call duration (sec)']):
#    if x < 80:
#        x = x - df['Call duration (sec)'].min()
#    df.loc[i, 'Call duration (sec)'] = x
plt.hist(1.1*(df['Call duration (sec)'])- min(df['Call duration (sec)']), bins=100)
plt.show()
```



In [16]:

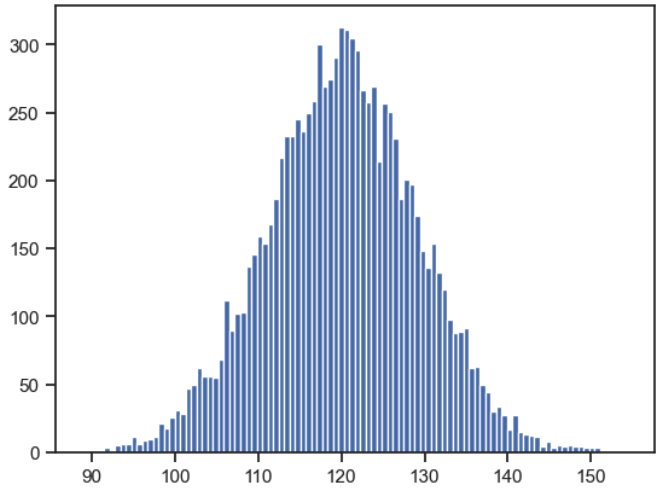
```
import scipy.stats as stats
from collections import Counter
expected_data = (stats.expon.rvs(scale = 109.84, size = 10000)/1.1) + min(df['Call duration (sec)'])
plt.hist(expected_data, bins= 100)
plt.show()
stats.kstest(df['Call duration (sec)'],expected_data)
```



Out[16]: KstestResult(statistic=0.0099, pvalue=0.7112026399498019, statistic_location=28.264200142512415, statistic_sign=1)

d) Lastly, for velocity we test using the Kolmogorov-Smirov test and get a p-value of 0.6042 as shown below which is greater than 0.05 and we don't reject the null hypothesis. Therefore we can model our input with a normal distribution with mu = 120.07 and standard deviation as 9.02.

In [225]:
```
import scipy.stats as stats
from collections import Counter
expected_data = stats.norm.rvs(loc= 120.07, scale = 9.02, size = 10000)
plt.hist(expected_data, bins= 100)
plt.show()
stats.kstest(df['velocity (km/h)'], expected_data)
```



Out[225]: KstestResult(statistic=0.0108, pvalue=0.6042251174586427)

Corrections/Changes to Flowchart

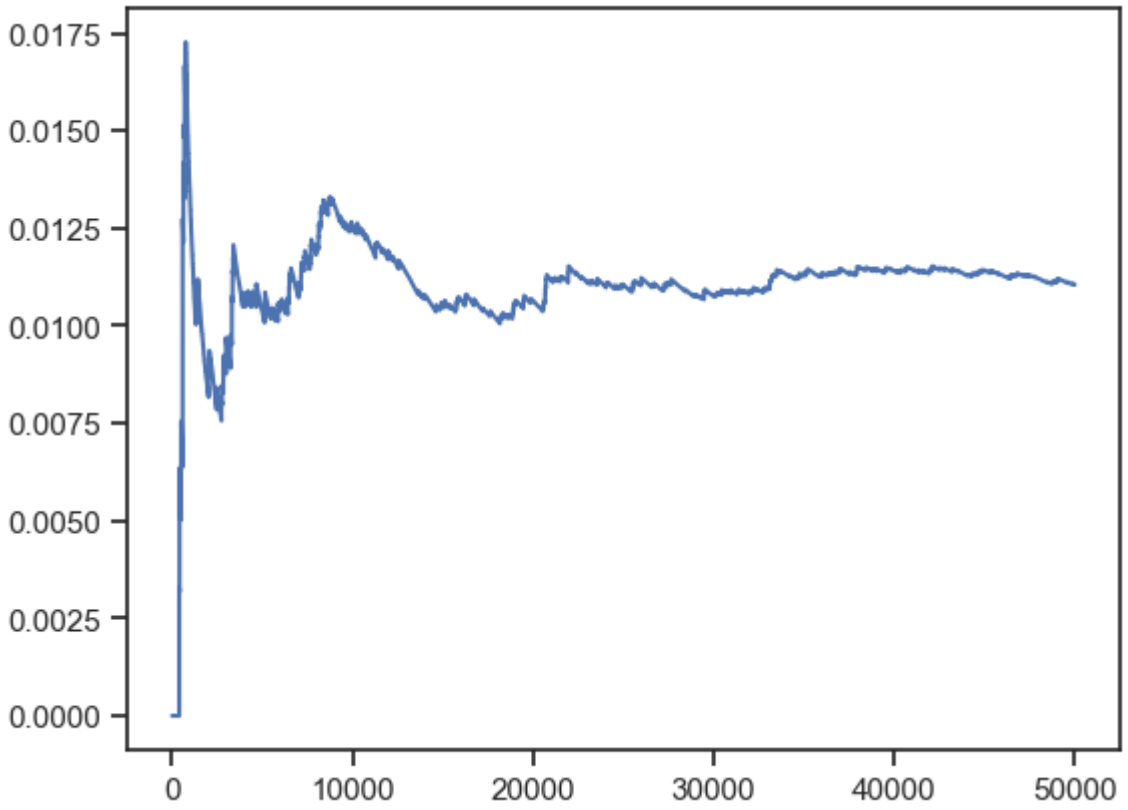
In my flowchart from Assignment 1 part 1, I made the following changes, some of the changes made are different from the feedback provided, since, the implementation follows that of my source code:

(a) Main Program: Generate an Initiation event and it to FEL
(b) Call Initiation Event:
    (i) Added an if statement for when the FCA scheme was no channel reserved vs when it had one reserved for handovers
        (1) When one call is reserved for handovers, if the cell channel free <=1 we block the call instead of utilising it for initiation.
    (ii) Number of blocked calls are not incremented in the initiation event.
    (iii) Create handover events whenever the base station is not busy. Continuation of recording the time of last event and storing it in the event. This makes it easier to compute how long the call has started in the handover process.
(c) Call Handover Event:
    (i) Removed the creation of call termination event when call is dropped.
    (ii) While we do not update duration rather calculate it from initiation time to the time of the event, we do update the position with each handover.
(d) Call Termination Event:
    (i) Updated queue to FEL

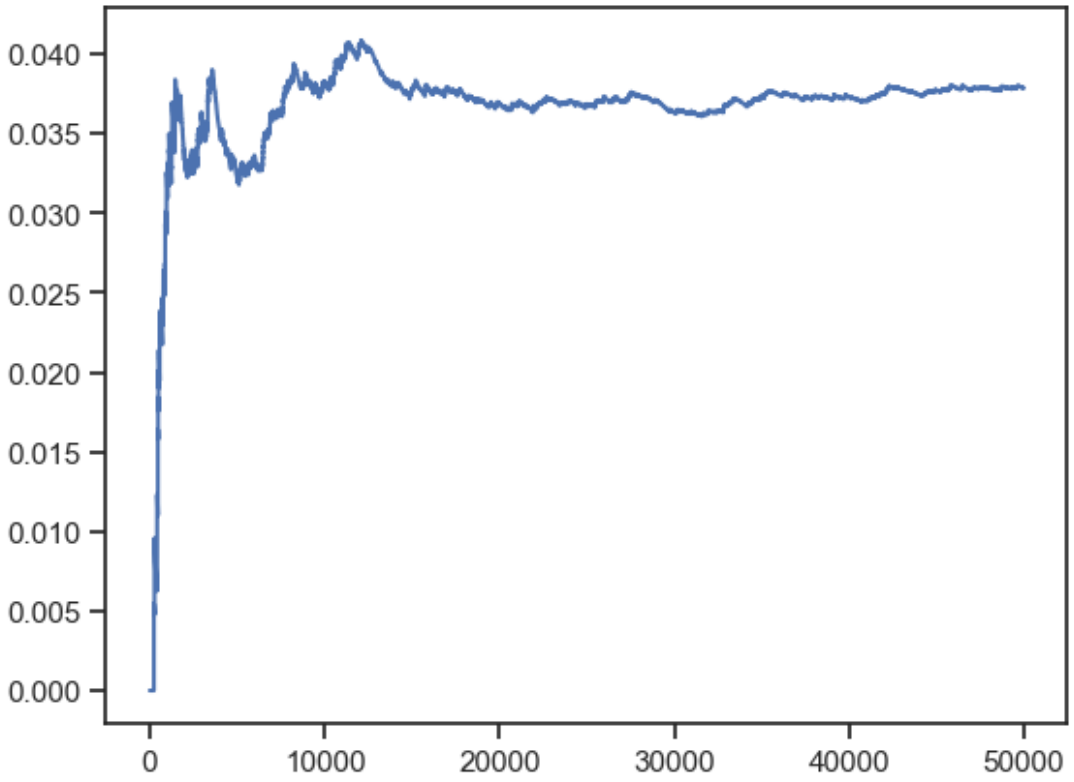
Warmup Period

1) For getting the warmup period we want our model to reach a steady state. In our simulation that means the metrics we want to take into account at the end of our simulation: percent of calls dropped and percent of calls blocked, should be roughly stable when we start to keep track of them.

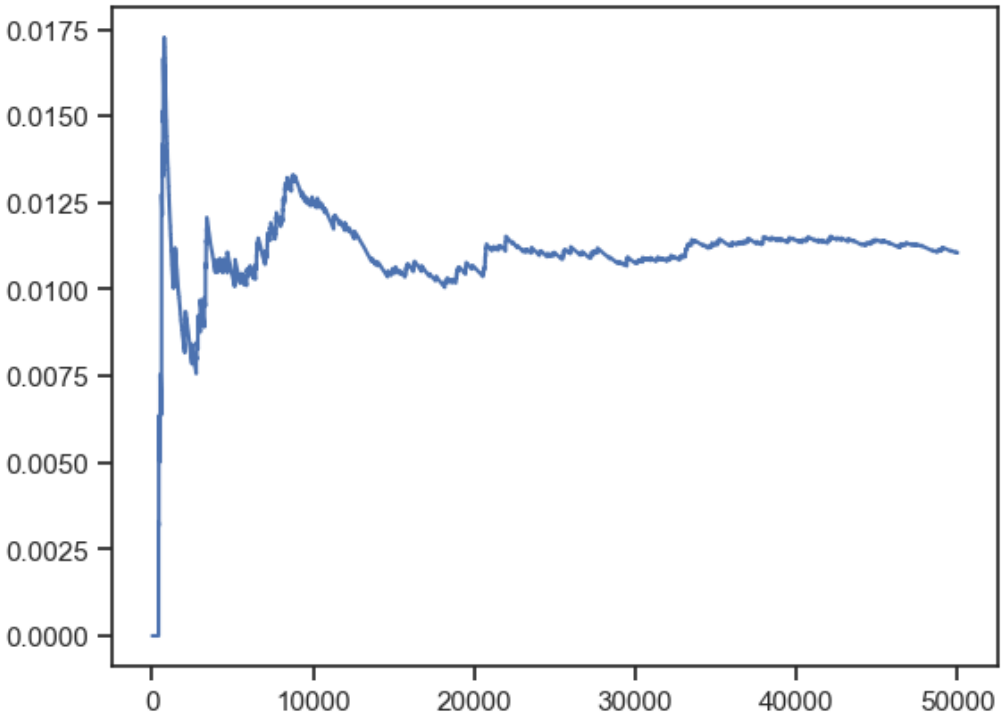
We can see when the metrics stabalise by looking at each one of them individually.



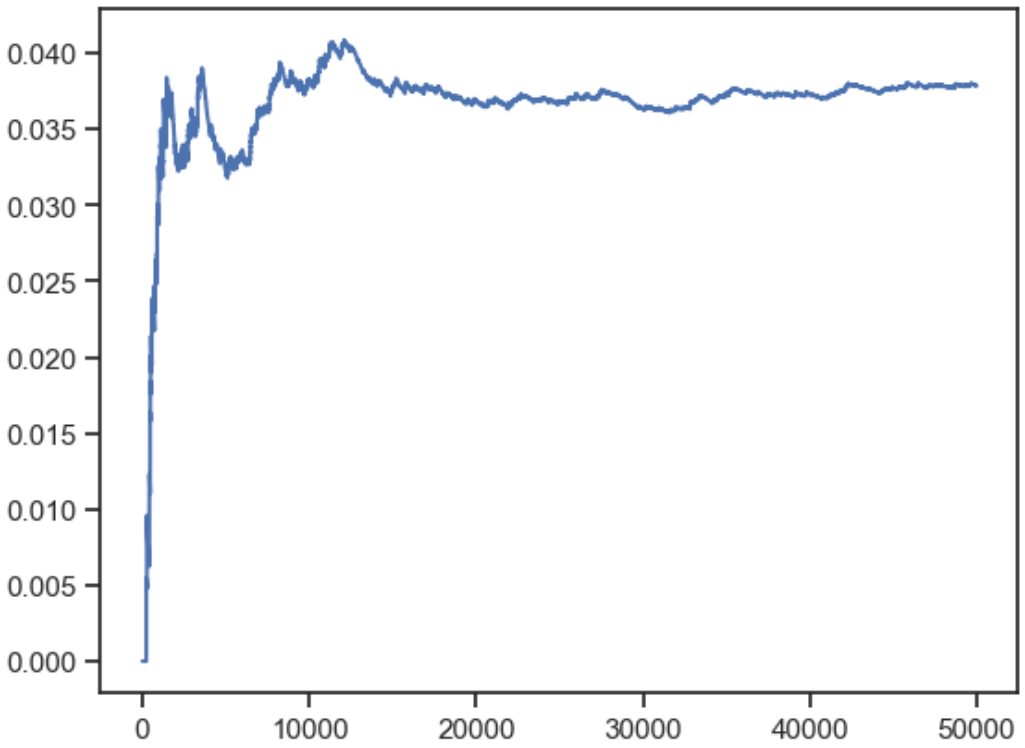
(a) Percent of Calls Dropped, FCA = no station reserved, stabalises at roughly clock = 32000 seconds



(b) Percent of Calls Blocked, FCA = no station reserved , stabalises at roughly clock = 34000 seconds



(c) Percent of Calls Dropped, FCA = 1 reserved for Handovers, the calls stabalise at roughly clock = 35000 seconds



(d) Percent of Calls Blocked, FCA = 1 reserved for Handovers, the calls seem to stabalise much earlier at roughly clock = 20000 seconds

We move forward with a warmup period of 35000 seconds, since after that time all metrics are stabalised.

Summary of Simulation Results, Conclusion & Recommendation:

% Dropped: 1.9419271314962085, % Blocked: 1.3778435361568337
% Dropped: 2.0402636092804913, % Blocked: 1.4173512683939693
% Dropped: 2.0599595365091043, % Blocked: 1.7288946109987127
% Dropped: 1.8046903627702455, % Blocked: 1.5390252839868084
% Dropped: 2.261785811608684, % Blocked: 1.8439458624761558
% Dropped: 1.9026909486273442, % Blocked: 1.4134275618374559
% Dropped: 1.762317770806422, % Blocked: 1.4855139324598634
% Dropped: 2.389789083008962, % Blocked: 1.4664614827554994
% Dropped: 2.2370343316289265, % Blocked: 1.5157048940832725
% Dropped: 1.8694170733953401, % Blocked: 1.3813426650704486
% Dropped: 2.033835629102339, % Blocked: 1.4144402329666268
% Dropped: 1.9711626209162256, % Blocked: 1.450994707063333
% Dropped: 2.0959985482261136, % Blocked: 1.4608474730060792
% Dropped: 1.9267822736030826, % Blocked: 1.6882282778236537
% Dropped: 1.8857561332845112, % Blocked: 1.2266569022336142
% Dropped: 2.496382054992764, % Blocked: 1.6009406657018814
% Dropped: 1.7049598832968638, % Blocked: 1.604668125455872
% Dropped: 2.138548711387315, % Blocked: 1.480533723268141
% Dropped: 2.2238559088402865, % Blocked: 1.53464436684433
% Dropped: 2.2449352071545903, % Blocked: 1.3871144369410477
% Dropped: 2.435711698659906, % Blocked: 1.6207895689967402
% Dropped: 2.0027186225645672, % Blocked: 1.5586769370185773
% Dropped: 2.4002157497303127, % Blocked: 1.7889248471772743
% Dropped: 1.93100862392201, % Blocked: 1.4435695538057742
% Dropped: 2.040082681765076, % Blocked: 1.6176867080075492
% Dropped: 2.2151029748283753, % Blocked: 1.6750572082379862
% Dropped: 2.000367039823821, % Blocked: 1.5140392732611487
% Dropped: 2.102867546654529, % Blocked: 1.3928083750568958
% Dropped: 2.1196203114920285, % Blocked: 1.4468712561054282
% Dropped: 2.107879234167894, % Blocked: 1.2886597938144329
% Dropped: 1.9748816891153986, % Blocked: 1.383327266108482
% Dropped: 1.9876690899052176, % Blocked: 1.656390908254348
% Dropped: 2.0898258478460128, % Blocked: 1.2832263978001834
% Dropped: 2.08694067255992, % Blocked: 1.5219174337009023
% Dropped: 1.910299003322259, % Blocked: 1.5780730897009967
% Dropped: 2.05776506413214, % Blocked: 1.6609762849497094
% Dropped: 2.308452348872721, % Blocked: 1.9311955447767897
% Dropped: 2.248510023478418, % Blocked: 1.5351273252663897
% Dropped: 2.1871867310671647, % Blocked: 1.7406361068076186
% Dropped: 2.1615779519048908, % Blocked: 1.6391966135278755
Average % Dropped: 2.0848599402369064, Average % Blocked: 1.5328845394383914, FCA: 10
% Dropped: 1.1646178020149736, % Blocked: 4.039190313337647
% Dropped: 1.0706045223811722, % Blocked: 3.3871712044300875
% Dropped: 1.0991007357616496, % Blocked: 3.9694795167590153

Summary: When FCA = no channels reserved, we see the % of calls dropped are on average: 2.08% while the % of calls blocked are on average: 1.53%. When FCA = 1 Channel reserved for handover calls, we see the % of calls dropped are on average:

1.07% while the % of calls blocked are on average: 3.8%. These are after taking into account of the warmup period.

Therefore, when no channels are reserved, the model is able to beat the Quality of Service Requirements with its performance on blocked calls. It missed the mark by 1.08% on average with dropped calls, while blocked calls meet the requirements on average since 1.53% < 2%. Following which we see that when 1 channel is reserved for handover calls: the model disappoints on both Quality of Service Requirements. Dropped calls on average falls to 1.07% which is above the required <1% by 0.07%, however, blocked calls shoot up to 3.8% on average which is way higher than the required <2%.

Recommendation for the telecommunication company XPhone would be to increase their number of channels rather than trying to change their FCA. Reserving one channel for handovers, despite helping rate of dropped calls, had an adverse effect on the percent of blocked calls. We would, therefore, recommend the company to increase the number of their channels in order to meet the service requirements. We go a step further and increase the channel allocations to 11 and rerun the simulation. We finally, recommend them the FCA scheme where one channel is reserved for handovers. This is because the dropped calls on average are 0.57% and blocked calls on average are 1.95%. There XPhone will be able to beat the Quality of Service requirements with 11 Channels and one reserved for handovers. The results for 11 Channels and FCA scheme no channel reserved and one

reserved for handover can be seen below:

```
% Dropped: 1.0848755583918315, % Blocked: 0.6928617011578084
% Dropped: 0.8725686238865662, % Blocked: 0.7362297764042901
% Dropped: 0.7972873900293255, % Blocked: 0.7056451612903225
% Dropped: 0.8399525244225328, % Blocked: 0.6938738245229618
% Dropped: 0.9033423667570009, % Blocked: 0.5962059620596206
% Dropped: 1.132554136087705, % Blocked: 0.7701368125396394
% Dropped: 1.2695840086439762, % Blocked: 0.8553934810012607
% Dropped: 1.1453179402602163, % Blocked: 0.7879787428990287
% Dropped: 0.8299520472150499, % Blocked: 0.6547399483585393
% Dropped: 1.2410369553226697, % Blocked: 0.7170435741864313
% Dropped: 0.935179242688182, % Blocked: 0.6692949481984046
% Dropped: 0.9850856195912355, % Blocked: 0.800957466396612
% Dropped: 0.8337864781584193, % Blocked: 0.7069059271343121
% Dropped: 0.9132007233273056, % Blocked: 0.8589511754068716
% Dropped: 0.9220839096357769, % Blocked: 0.5255878284923928
% Dropped: 1.0901162790697674, % Blocked: 0.8266715116279071
% Dropped: 1.186970923813029, % Blocked: 0.6164887743835112
% Dropped: 0.8639505274645326, % Blocked: 0.7002546380502
% Dropped: 0.9027266028002948, % Blocked: 0.6448047162859248
% Dropped: 1.1012400518230614, % Blocked: 0.518230612622617
% Dropped: 0.865488640461594, % Blocked: 0.7843490804183195
% Dropped: 1.1890820646788578, % Blocked: 0.7656967840735069
% Dropped: 1.0446947674418605, % Blocked: 0.8902616279069767
% Dropped: 0.8780664433782926, % Blocked: 0.7694396668778855
% Dropped: 1.042223409941208, % Blocked: 0.9709602707999287
% Dropped: 1.27138022500686, % Blocked: 0.8231958291411323
% Dropped: 0.9587020648967551, % Blocked: 0.765117994100295
% Dropped: 1.0549093859886394, % Blocked: 0.6942566044540619
% Dropped: 0.7644007644007643, % Blocked: 0.9100009100009101
% Dropped: 1.1103973570707535, % Blocked: 0.5781407726897311
% Dropped: 1.1752915451895043, % Blocked: 0.7197521865889213
% Dropped: 0.7768625982777986, % Blocked: 0.7300636465743168
% Dropped: 0.8380917296002948, % Blocked: 0.5525879535826119
% Dropped: 0.9935004642525533, % Blocked: 0.659238625812442
% Dropped: 1.070488977592307, % Blocked: 0.7348271795337022
% Dropped: 1.116398243045388, % Blocked: 0.7320644216691069
% Dropped: 1.2074643249176729, % Blocked: 0.7317965605561654
% Dropped: 0.979858464888405, % Blocked: 0.7076755579749592
% Dropped: 1.0634362862955429, % Blocked: 0.6473090438320696
% Dropped: 1.06951871657754, % Blocked: 0.6979062811565304
Average % Dropped: 1.008217727562135, Average % Blocked: 0.7241002375632989,
FCA: 10
% Dropped: 0.5868329359985329, % Blocked: 1.78800660187053
% Dropped: 0.7145474532795896, % Blocked: 2.0795163063393183
% Dropped: 0.4981090305322387, % Blocked: 1.835624019924361
```

% Dropped: 0.7323324789454413, % Blocked: 1.9681435371658733
% Dropped: 0.48833423765599565, % Blocked: 1.6910833785494666
% Dropped: 0.4770204568388221, % Blocked: 1.9539491789744061
% Dropped: 0.4416229643941485, % Blocked: 1.766491857576594
% Dropped: 0.4304294937774867, % Blocked: 1.534574716945822
% Dropped: 0.43757564472581695, % Blocked: 1.6758216180988734
% Dropped: 0.5958291956305859, % Blocked: 2.148596190304234
% Dropped: 0.6806425265450585, % Blocked: 2.2234322533805244
% Dropped: 0.5130554283096657, % Blocked: 1.9514429683921208
% Dropped: 0.5973715651135006, % Blocked: 1.8013050271114786
% Dropped: 0.9162659892951102, % Blocked: 2.4675678127551484
% Dropped: 0.5874793464292271, % Blocked: 2.1020745364420783
% Dropped: 0.561136754457417, % Blocked: 2.072585754366911
% Dropped: 0.6821282401091405, % Blocked: 2.1646202819463394
% Dropped: 0.4506990434142752, % Blocked: 2.0235467255334805
% Dropped: 0.5769588684161548, % Blocked: 2.1775544388609713
% Dropped: 0.5201679138528928, % Blocked: 1.6426355174301879
% Dropped: 0.5311355311355311, % Blocked: 1.684981684981685
% Dropped: 0.6426735218508998, % Blocked: 2.065736320235035
% Dropped: 0.48304775792927457, % Blocked: 1.8775063798760483
% Dropped: 0.8176100628930818, % Blocked: 2.120395327942498
% Dropped: 0.619138668851862, % Blocked: 2.0395156150414278
% Dropped: 0.5403425222089935, % Blocked: 1.9690447843209085
% Dropped: 0.6089801854208327, % Blocked: 1.8451190692601345
% Dropped: 0.49792531120331945, % Blocked: 2.056247118487782
% Dropped: 0.6149609912804038, % Blocked: 2.276273519963286
% Dropped: 0.5721031601888849, % Blocked: 1.5346894297130405
% Dropped: 0.49586776859504134, % Blocked: 1.6620752984389346
% Dropped: 0.5965491923641704, % Blocked: 1.8997797356828192
% Dropped: 0.4108088369545372, % Blocked: 1.8075588825999633
% Dropped: 0.6057866184448463, % Blocked: 2.115732368896926
% Dropped: 0.5288110867979576, % Blocked: 2.0605397520058353
% Dropped: 0.49656915854098954, % Blocked: 2.058504875406284
% Dropped: 0.5737022300360878, % Blocked: 1.9894512815767558
% Dropped: 0.48820928518791457, % Blocked: 1.9252026529108328
% Dropped: 0.6038647342995169, % Blocked: 2.127461910070606
% Dropped: 0.6994294128474139, % Blocked: 1.7669795692987302
Average % Dropped: 0.5732540923019184, Average % Blocked: 1.9492927577755452, FCA: 9