1.

1. A regression problem, a prediction problem, 50000 observations, 5 predictors
2. A classification problem, a inference problem, 85 observations, 9 predictors

2a.

```
data = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.csv')
data.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

2b.

```
[52] (data.alive == 'yes').mean()

0.3838383838383838
```

38% of passengers survived

2c.

```
[122] data[(data.pclass == 3)&(data.alive == 'yes')].age.aggregate([np.min,np.max])

amin     0.42
amax    63.00
Name: age, dtype: float64
```

The age of the youngest and oldest 3rd class passengers survived is 0.42 and 63 years old.

2d.Average fare and passengers. (There are 2 missing values in embark town)

```
data.pivot_table(index = 'class',columns = 'embark_town',values = 'fare',aggfunc = [np.mean,'count'])
```

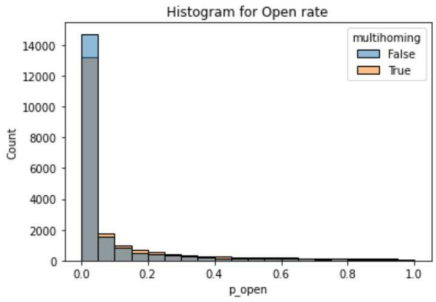
	mean			count		
embark_town	Cherbourg	Queenstown	Southampton	Cherbourg	Queenstown	Southampton
class						
First	104.718529	90.000000	70.364862	85	2	127
Second	25.358335	12.350000	20.327439	17	3	164
Third	11.214083	11.183393	14.644083	66	72	353

3a.

To compare the distribution of p-open rates and multihoming status, I randomly select 20000 samples from both classes and make a histogram for each with different colors. The distribution

is largely the same for both classes. Therefore, I conclude that p_open rate does not depend on multihoming status.

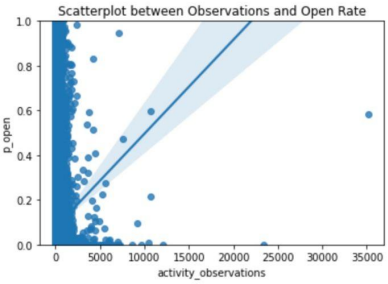
```
df = pd.read_csv('cogo-all.tsv', delimiter='\t')
df['multihoming']  = (df.browser1+df.browser2+df.browser3>1)
x = df[df.multihoming == False].sample(20000)
y = df[df.multihoming == True].sample(20000)
newdf = pd.concat([x,y])
newdf.reset_index(inplace = True,drop = True)
sns.histplot(x = 'p_open',hue = 'multihoming',data = newdf,bins = 20)
plt.title('Histogram for Open rate')
plt.show()
```



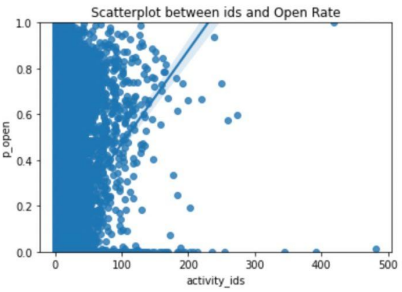
3b.

To draw more analysis on the dataset, I have done some exploratory data analysis on the dataset, and made some scatter plots between open rate and different variables, and found out that ids and observations are largely related to open rate.

```
sns.regplot(data = df,x = 'activity_observations',y = 'p_open')
plt.ylim(0, 1)
plt.title('Scatterplot between Observations and Open Rate')
plt.show()
```




```
[247] sns.regplot(data = df,x = 'activity_ids',y = 'p_open')
      plt.ylim(0, 1)
      plt.title('Scatterplot between ids and Open Rate')
      plt.show()
```



As shown in the trend line, the more times the user has been seen online, the higher the open rate. The more unique computers (inlc. phones, tablets) have we seen the user, the higher the open rate. Therefore, we should target customers with more unique devices and surf online more frequently.

4a.

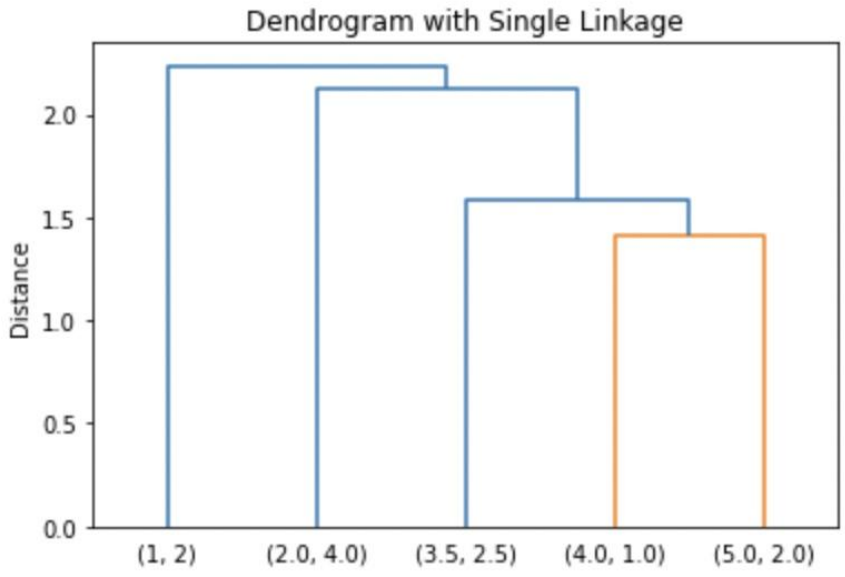
```
[262] df = [(1, 2), (2.0, 4.0), (3.5, 2.5), (4.0, 1.0), (5.0, 2.0)]
      name = ['(1, 2)','(2.0, 4.0)','(3.5, 2.5)','(4.0, 1.0)','(5.0, 2.0)']
```

```
np.array(["{:.2f}".format(((x1-y1)**2+(x2-y2)**2)**(1/2))for x1,x2 in df for y1,y2 in df]).reshape(5,-1)
```

```
array([['0.00', '2.24', '2.55', '3.16', '4.00'],
       ['2.24', '0.00', '2.12', '3.61', '3.61'],
       ['2.55', '2.12', '0.00', '1.58', '1.58'],
       ['3.16', '3.61', '1.58', '0.00', '1.41'],
       ['4.00', '3.61', '1.58', '1.41', '0.00']], dtype='<U4')
```

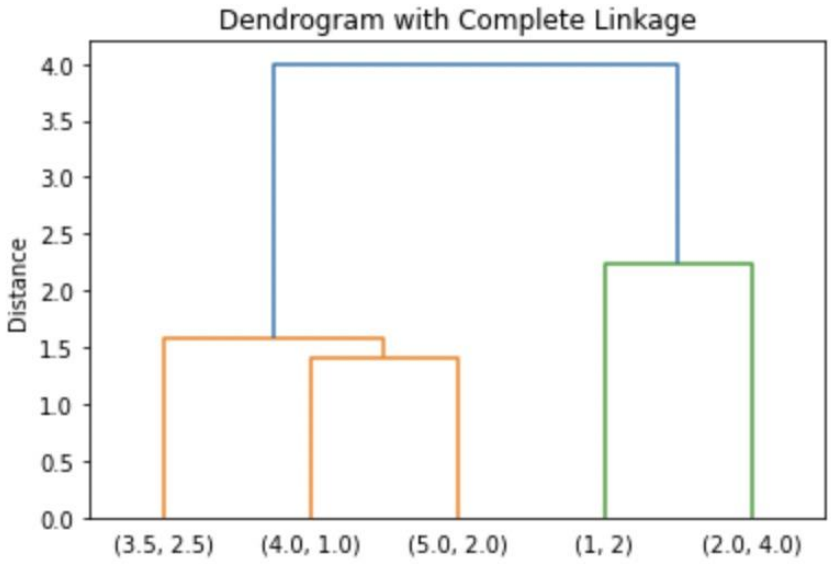
4b

```
[80] merging1 = linkage(df,method='single',metric='euclidean')
     dendrogram(merging1,labels=name,leaf_font_size=10)
     plt.title('Dendrogram with Single Linkage')
     plt.ylabel('Distance')
     plt.show()
```



4c

```
[81] merging2 = linkage(df,method='complete',metric='euclidean')
     dendrogram(merging2,labels=name,leaf_font_size=10)
     plt.title('Dendrogram with Complete Linkage')
     plt.ylabel('Distance')
     plt.show()
```



4d.

```
[90] cluster1, cluster2 = [], []
     for idx,x in enumerate(df):
       if fcluster(merging1,2.2,criterion='distance')[idx] == 1:
         cluster1.append(x)
       else:
         cluster2.append(x)
     print('Single cluster1:',cluster1,'Single cluster2:',cluster2)
```

```
Single cluster1: [(2.0, 4.0), (3.5, 2.5), (4.0, 1.0), (5.0, 2.0)] cluster2: [(1, 2)]
```

```
[91] cluster1, cluster2 = [], []
     for idx,x in enumerate(df):
       if fcluster(merging2,3,criterion='distance')[idx] == 1:
         cluster1.append(x)
       else:
         cluster2.append(x)
     print('Complete cluster1:',cluster1,'Complete cluster2:',cluster2)
```

```
Complete cluster1: [(3.5, 2.5), (4.0, 1.0), (5.0, 2.0)] Complete cluster2: [(1, 2), (2.0, 4.0)]
```

5a

```
[290] initia = np.array([df[3],df[4]])
      for iters in range(1,11):
        clusters = KMeans(n_clusters=2, init=initia,n_init=1,max_iter=iters,random_state = 0).fit(df)
        print('The centriod for current iteration is:',clusters.cluster_centers_.tolist())
        print('The current assignment to each cluster is:',clusters.labels_)
        if iters == 1:
          old = clusters.cluster_centers_
        elif iters != 1:
          if [x1 for x in old for x1 in x] == [x2 for y in clusters.cluster_centers_ for x2 in y]:
            break
          else:
            old = clusters.cluster_centers_
```

```
The centriod for current iteration is: [[2.625, 2.375], [5.0, 2.0]]
The current assignment to each cluster is: [0 0 0 1 1]
The centriod for current iteration is: [[2.166666666666667, 2.833333333333333], [4.5, 1.5]]
The current assignment to each cluster is: [0 0 0 1 1]
The centriod for current iteration is: [[2.166666666666667, 2.833333333333333], [4.5, 1.5]]
The current assignment to each cluster is: [0 0 0 1 1]
```

5b

```
initia = np.array([df[0],df[4]])
for iters in range(1,11):
  clusters = KMeans(n_clusters=2, init=initia,n_init=1,max_iter=iters,random_state = 0).fit(df)
  print('The centriod for current iteration is:',clusters.cluster_centers_.tolist())
  print('The current assignment to each cluster is:',clusters.labels_)
  if iters == 1:
    old = clusters.cluster_centers_
  elif iters != 1:
    if [x1 for x in old for x1 in x] == [x2 for y in clusters.cluster_centers_ for x2 in y]:
      break
    else:
      old = clusters.cluster_centers_
```

```
The centriod for current iteration is: [[1.5, 3.0], [4.166666666666666, 1.8333333333333335]]
The current assignment to each cluster is: [0 0 1 1 1]
The centriod for current iteration is: [[1.5, 3.0], [4.166666666666666, 1.8333333333333335]]
The current assignment to each cluster is: [0 0 1 1 1]
```

For question 5, sklean.Kmeans is giving the cluster assignment of each data point after the iteration has been completed instead of the cluster assignment that forms the centroids.