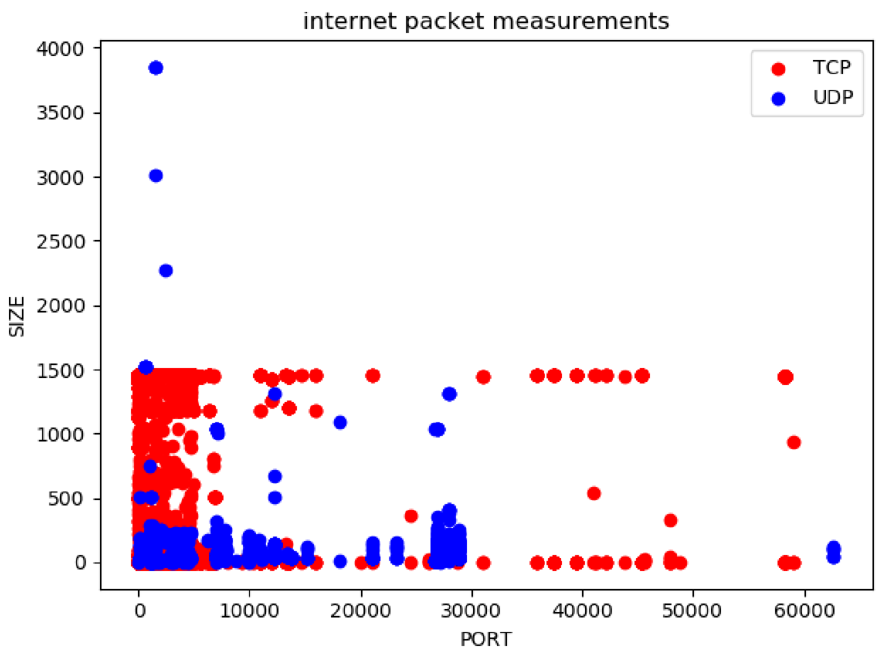
Name: Chaoyang Zhu   UIN: 930001840

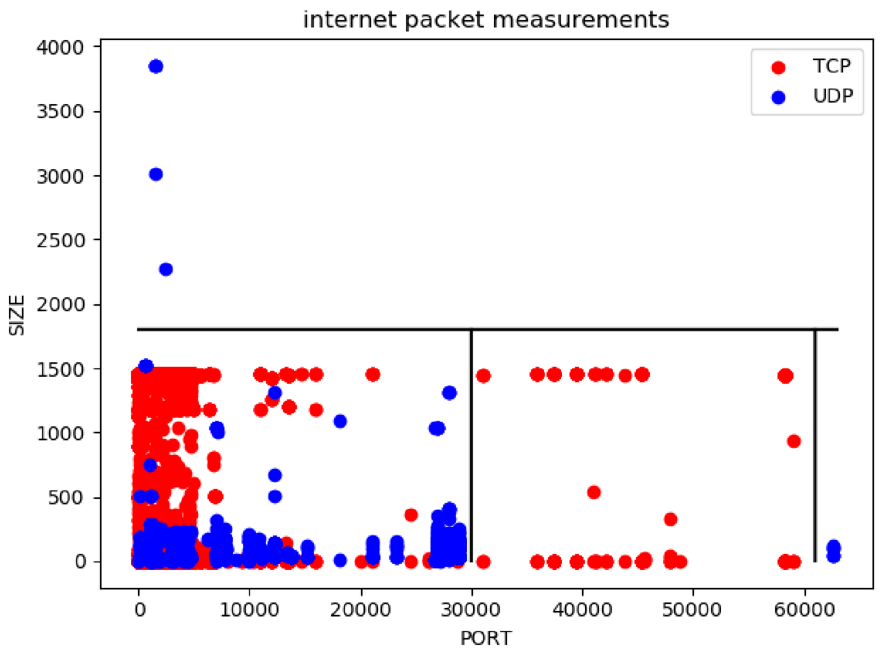
1.

Linear axis scaling:

```
import pandas as pd
import matplotlib.pyplot as plt
import math
def main():

    df = pd.read_excel('data.xlsx')
    port = df['PORT']
    size = df['SIZE']
    protocol = df['CLASS']

    port1 = []
    port2 = []
    size1 = []
    size2 = []

    for i in range(len(protocol)):
        x = [port[i], size[i]]
        data.append(x)
        if protocol[i] == 'tcp':
            port1.append(port[i])
            size1.append(size[i])
        else:
            port2.append(port[i])
            size2.append(size[i])

    plt.figure()
    plt.scatter(port1, size1, c='red')
    plt.scatter(port2, size2, c='blue')

    plt.title('internet packet measurements')
    plt.xlabel('PORT')
    plt.ylabel('SIZE')
    plt.legend(['TCP', 'UDP'])

if __name__ == '__main__':
    main()
```

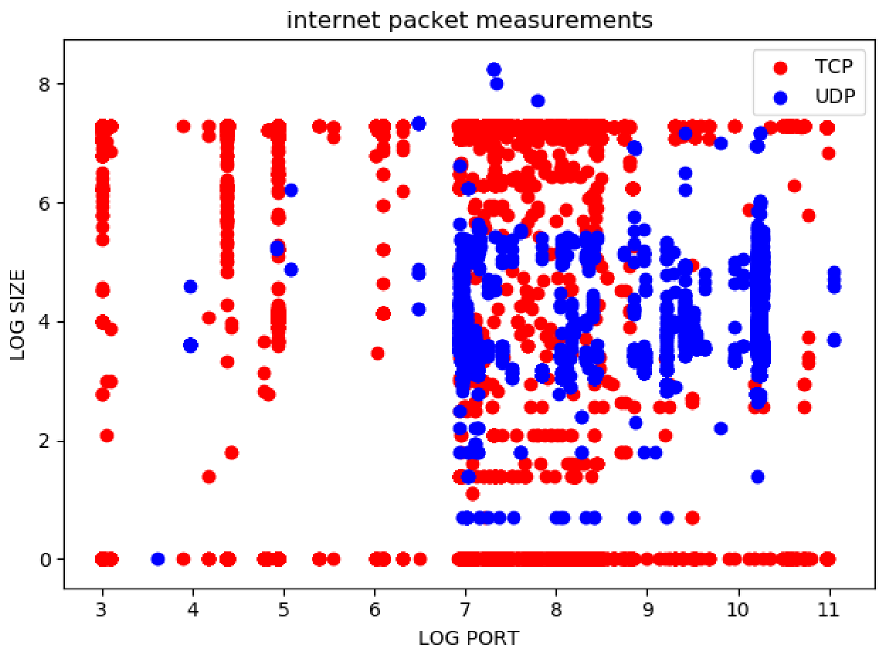


Without doing any computation, I can only separate data by **3 split points**:
SIZE<=1800, PORT>=61000, SIZE<=30000
The graph is as follows:

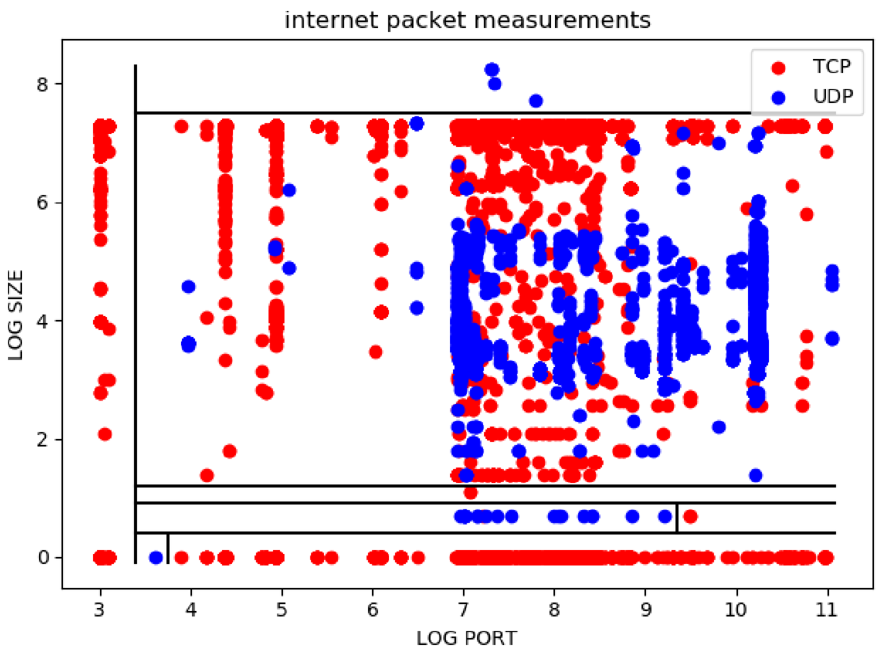


Log axis scaling:

```
for i in range(len(size1)):
    if size1[i] > 0:
      logport1.append(math.log(port1[i]))
      logsize1.append(math.log(size1[i]))
    else:
      logport1.append(math.log(port1[i]))
      logsize1.append(size1[i])

  for i in range(len(size2)):
    if size2[i] > 0:
      logport2.append(math.log(port2[i]))
      logsize2.append(math.log(size2[i]))
    else:
      logport2.append(math.log(port2[i]))
      logsize2.append(size2[i])

  plt.figure()
  plt.scatter(logport1, logsize1, c='red')
  plt.scatter(logport2, logsize2, c='blue')

  plt.title('internet packet measurements')
  plt.xlabel('LOG PORT')
  plt.ylabel('LOG SIZE')
  plt.legend(['TCP', 'UDP'])

  plt.show()
```



Without doing any computation, I can only separate data by **7 split points**:
LOG PORT<=3.4, LOG SIZE>=7.5, LOG SIZE<=0.4,
LOG PORT<=3.75, LOG SIZE<=1.19, LOG SIZE<=0.9,
LOG PORT>=9.35
The graph is as follows:



2.
Choose **sklearn.tree.DecisionTreeClassifier** in python as the package for desicion tree analysis.

way to **compute prediction accuracy** (using the full dataset for both training and testing):

```
from sklearn import tree
   clf = tree.DecisionTreeClassifier()
   clf = clf.fit(data, protocol)
   predict_results = clf.predict(data)
   scores = clf.score(data, protocol)
```

way to **generate graphical representations** of decision trees:

```
import graphviz
   clf = tree.DecisionTreeClassifier()
   clf = clf.fit(data, protocol)
   dot_data = tree.export_graphviz(clf, out_file=None)
   graph = graphviz.Source(dot_data)
   graph.render("TCP UDP")
```

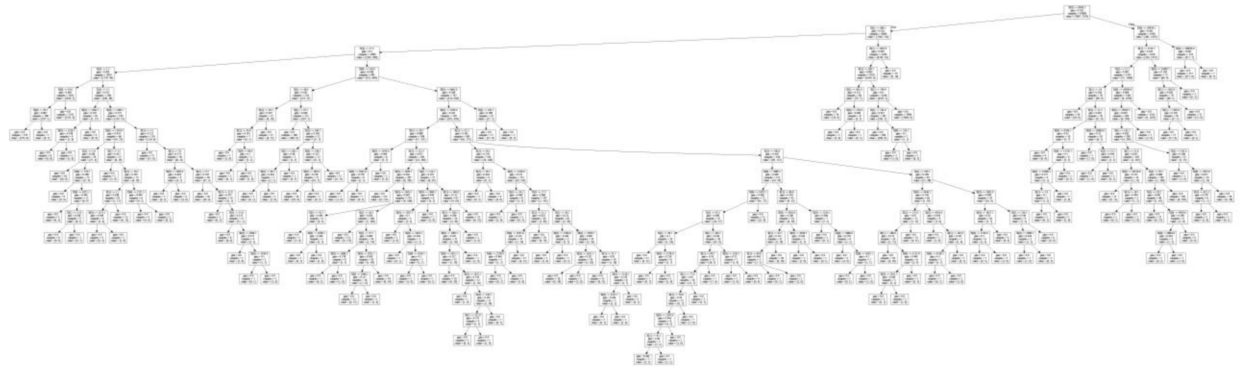
3.
Create a decision tree for predicting CLASS from PORT and SIZE.

```
   for i in range(len(protocol)):
      x = [port[i], size[i]]
```

```
        data.append(x)
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(data, protocol)
```

4.
Create a graph of the decision tree.

```
import graphviz
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(data, protocol)
    dot_data = tree.export_graphviz(clf, out_file=None)
    graph = graphviz.Source(dot_data)
    graph.render("TCP UDP")
```



5.
The split points chosen by computer is more precise than my choice.

6.
Compute **the prediction accuracy for the full dataset**

```
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(data, protocol)
    predict_results = clf.predict(data)
    scores = clf.score(data, protocol)
```

The prediction accuracy for the full dataset is **0.9998**.

7.
Training: the first 7000 records.
Testing: the last 3000 records.

```
    for i in range(len(protocol)):
        if i < 7000:
            datafirst.append(data[i])
            protocolfirst.append(protocol[i])
        else:
            datalast.append(data[i])
        protocollast.append(protocol[i])

    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(datafirst, protocolfirst)
    predict_results = clf.predict(datalast)
    scores = clf.score(datalast, protocollast)
```

The prediction accuracy for the last 3000 records is **0.6683333333333333.**

8.
The prediction accuracy for the full data is higher than the split way in item 7.
For the full data, the prediction accuracy is close to 1.
Since only partial data were used to construct the decision tree, the prediction accuracy is less than full data.
The last 3000 records contain some information that is not contained in the first 7000 records.

9.
A **random** split of the data (70% training, 30% testing)

```
    feature_train, feature_test, target_train, target_test \
        = train_test_split(data, protocol, train_size=0.7, test_size=0.3)
    clf = tree.DecisionTreeClassifier()
    clf.fit(feature_train, target_train)
    predict_results = clf.predict(feature_test)
    scores = clf.score(feature_test, target_test)
```

The prediction accuracy of the random split of data (70% training, 30% testing) is **about 0.99** which is much more than before.