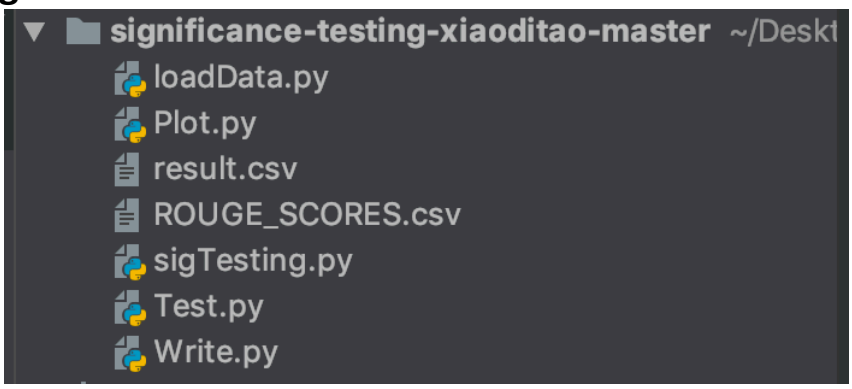
Report for hw4

Name: Xiaodi Tao AndrewID: xiaodit

Part 1 – The design of the code



1. There are 5 python files and 2 csv files.
   - loadData.py: It load the data from the csv named “ROUGE_SCORES.csv”.
   - Plot.py: It help to plot box plot and violin plot of the data from “ROUGE_SCORES.csv”.
   - sigTesting.py: It is the main class which has the main method to call methods in other python files.
   - Test.py: It has the method to do ksTest, t Test and wilcoxonTest.
   - Write.py: It write out the result to “results.csv”
2. Here are some screenshots for each of the python files.

   loadData.py:

```
from numpy import genfromtxt


class LoadData(object):
    def __init__(self, filePath):
        self.filePath = filePath


    def loadData(self):
        self.data = genfromtxt(self.filePath, delimiter=',')[1:].T
        mean = self.data.mean(axis=1)
        return mean,self.data
```

   Plot.py

```
import matplotlib.pyplot as plt


class Plot(object):
    def __init__(self,data):
        self.data = data


    def twoPlot(self):
        plt.figure()
        plt.boxplot(self.data.T)
        plt.show()
        plt.violinplot(self.data.T)
        plt.show()
```

   sigTesting.py:

```
class SignificanceTesting(object):
    def __init__(self):
        pass


if __name__ == '__main__':
    filePath = "ROUGE_SCORES.csv"
    loadData = LoadData(filePath)
    meanArr, data = loadData.loadData()
    test = Test()
    write = Write(data)
    write.writeOutput(meanArr, 4)
    plot = Plot(data)
    plot.twoPlot()
```

Test.py

```
    def ksTest(self, listA, listB):
        value, pvalue = ttest_ind(listA, listB)
        return pvalue


    def tTest(self, listA, listB):
        value, pvalue = ks_2samp(listA, listB)
        return pvalue


    def wilcoxonTest(self, listA, listB):
        value, pvalue = wilcoxon(listA, listB)
        return pvalue
```

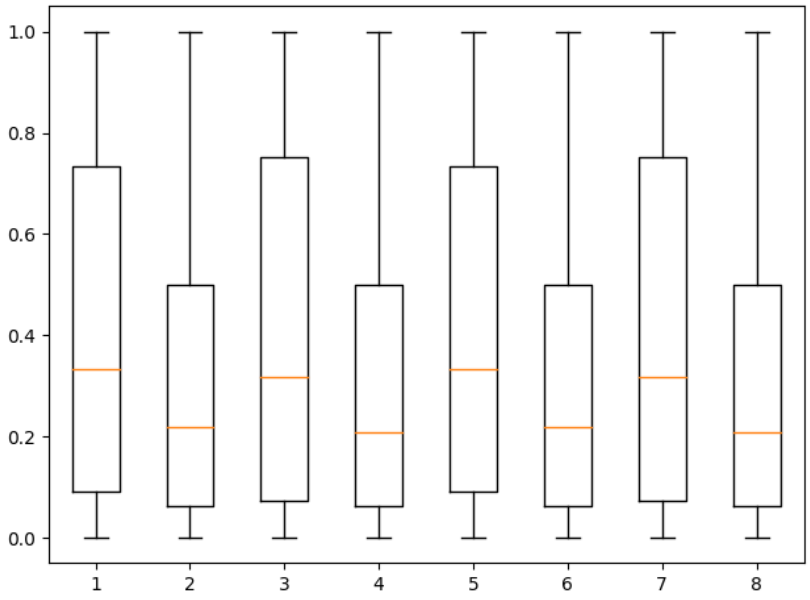
Write.py

```
resultsData = [[0 for x in range(w)] for y in range(h)]

resultsData[0] = ['metric & model', 'mean diff', 'P(T test)', 'P(wilcoxon test)', 'P(ks test)']
for row in range(1,13):

    resultsData[row][0] = dictScoreCsv.get(listAll[row-1][0]) + ' ' + dictScoreCsv.get(listAll[row-1][1])
    resultsData[row][1] = meanArr[listAll[row-1][0]] - meanArr[listAll[row-1][1]]
    resultsData[row][2] = ttest_ind(self.data[listAll[row-1][0]],self.data[listAll[row-1][1]]).pvalue
    resultsData[row][3] = wilcoxon(self.data[listAll[row-1][0]],self.data[listAll[row-1][1]]).pvalue
    resultsData[row][4] = ks_2samp(self.data[listAll[row-1][0]],self.data[listAll[row-1][1]]).pvalue

with resultsFile:
    writer = csv.writer(resultsFile)
    writer.writerows(resultsData)
resultsFile.close()
```

3. Details about the code design.
   When we input the name of the input data csv file in the main method of the main class, it will use method in loadData class to load data. Then we call test methods in Test class to do the three test on each combination. The code in Write class is **not** hardcode. When we input how many test we want to do, it will automatically set up the rows and colomns of the output files, and loop to insert the p-value into the output file. Also, we try to plot the box plot as well as violin plot to check the distribution of the input data.

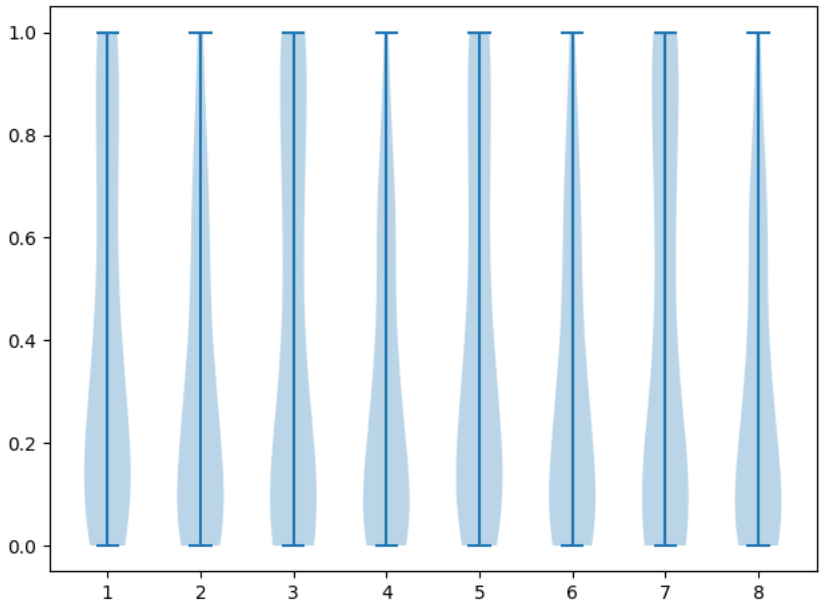
Part 2 – Analysis of the result

1. Firstly, let's see the boxplot of each of the performance in different conditions:



In the boxplot, we can find that condition 1,3,5,7 seem to have higher maximum scores and mean scores, but among condition 1,3,5,7, we cannot figure out which one is better. Thus, we try to plot violin plot.

2.

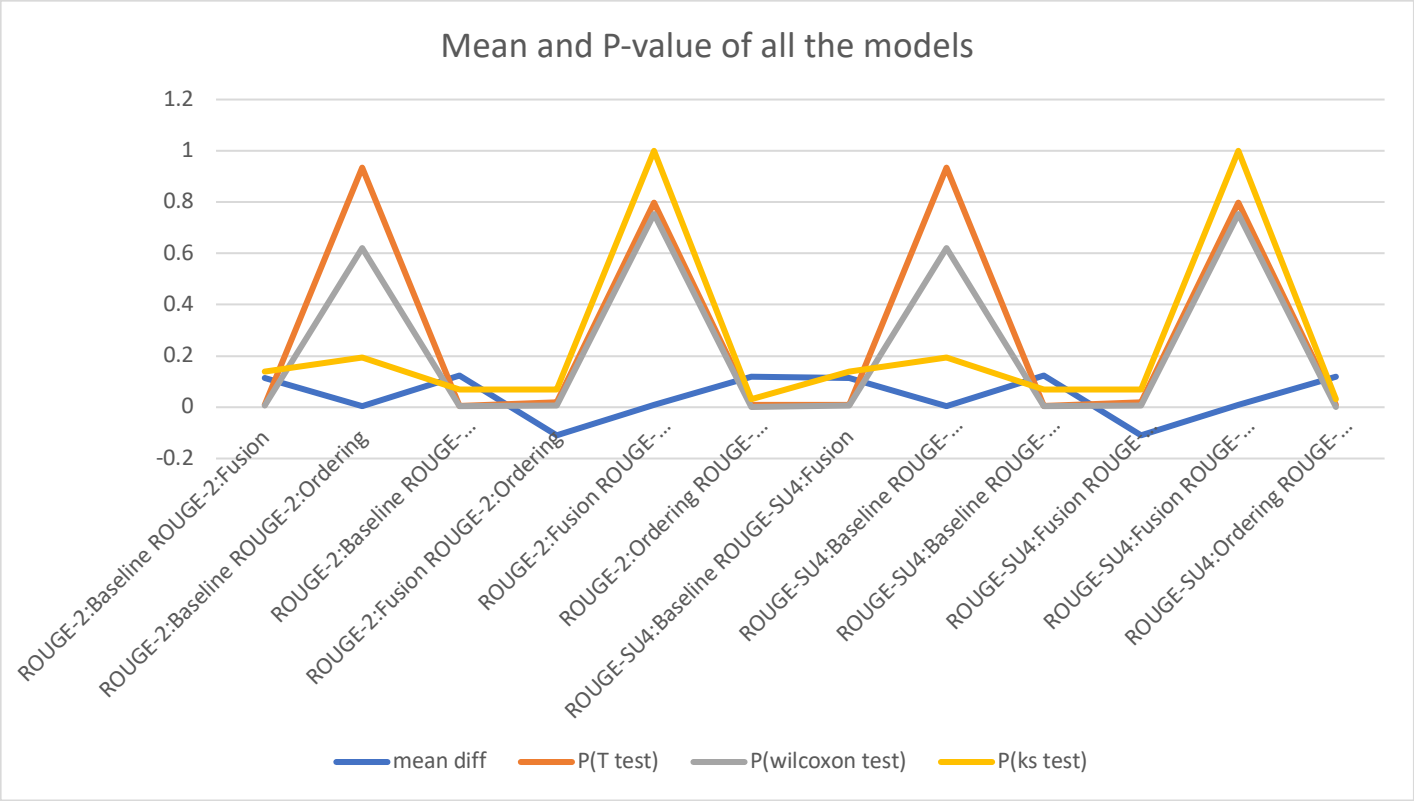


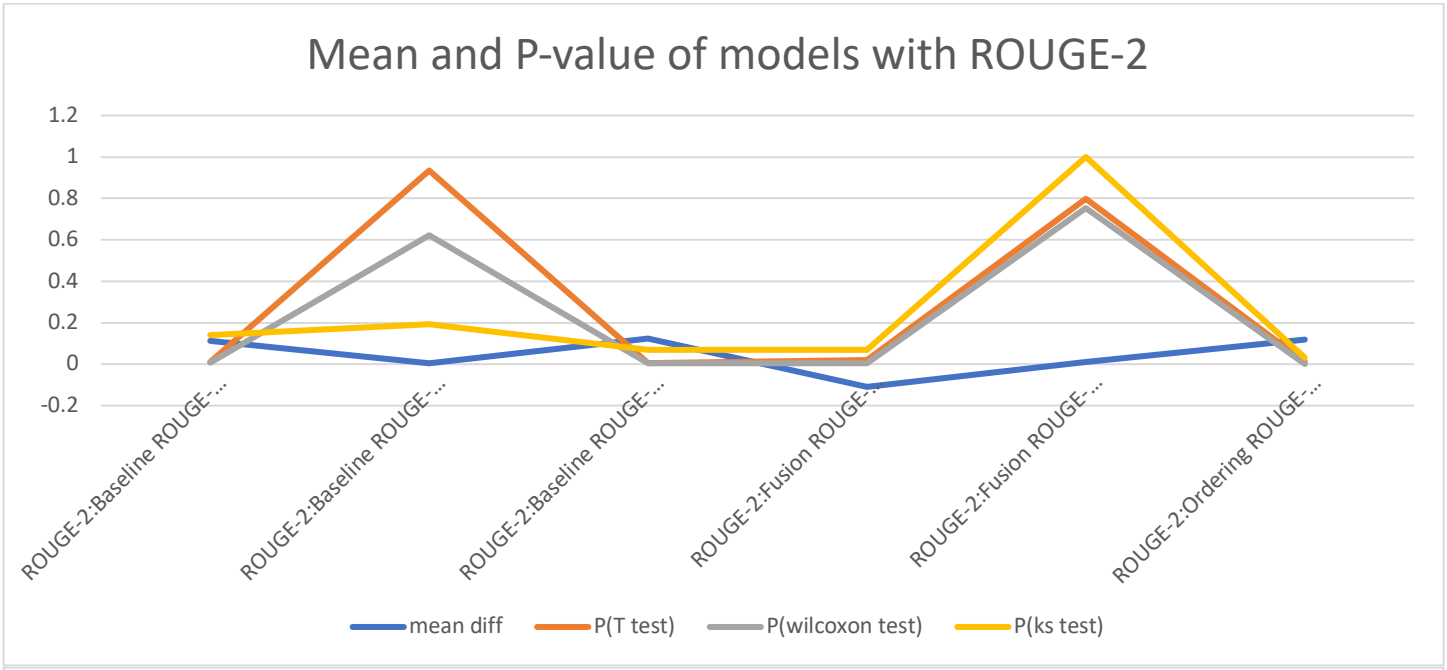
The violin plot also shows the mean, interquartile range. Also, the shading area shows the 95% confidence interval. we can also find that condition 1,3,5,7 seem to have higher scores in the 95% confidence interval and mean scores, but among condition 1,3,5,7, we cannot figure out which one is better. Thus, let's do T test, Wilcoxon test and ks test to see whether the conditions are significantly different with the base model.

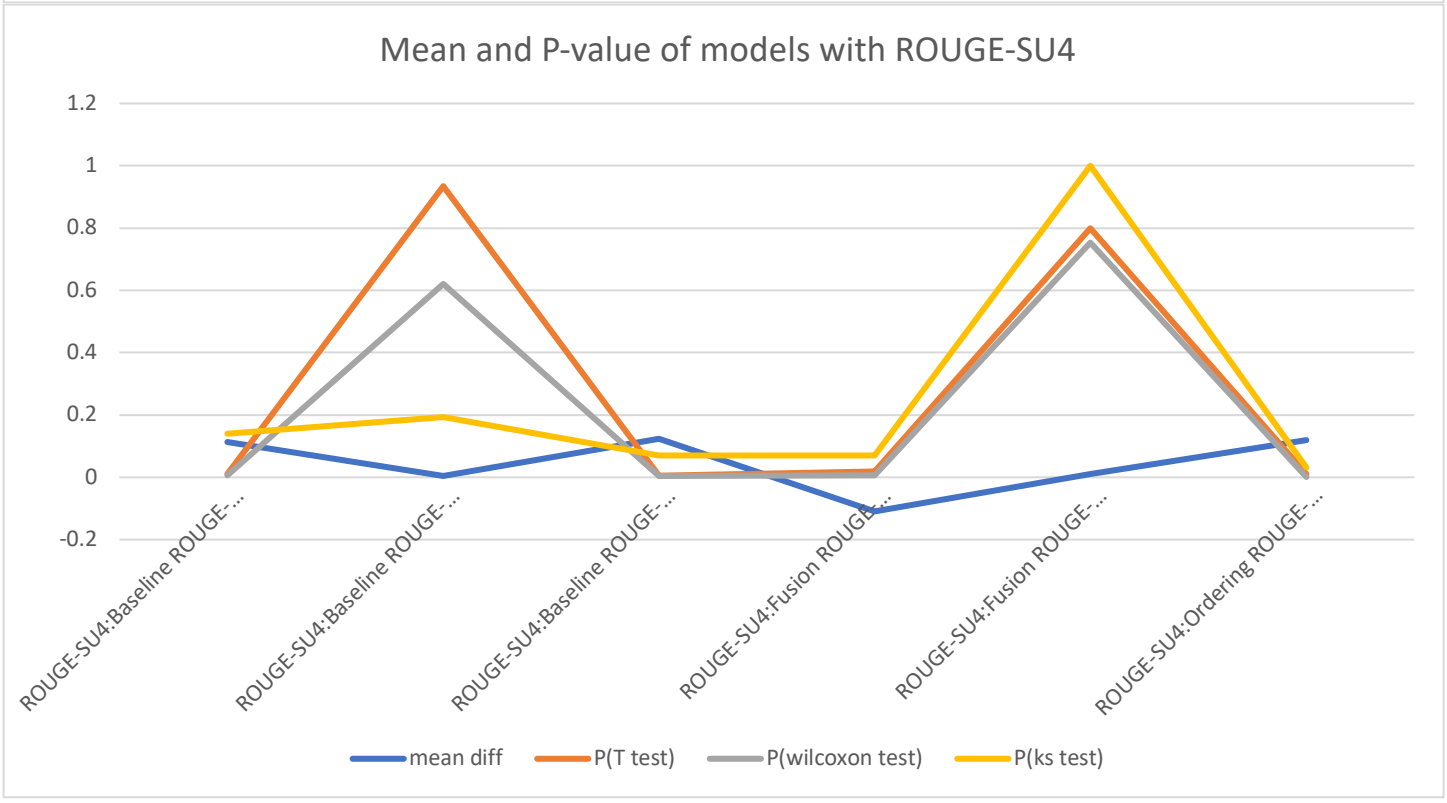
3. This is the result.csv file.

From the boxplot and output result, we can find that under each metric(ROUGE-2 and ROUGE-SU4), the Ordering condition has max value higher than the base model, the mean diff of these two models is almost zero, and the p-value is bigger than 0.05, so the two model is significantly different and the Ordering model performs better than base model. Thus, I recommend the Ordering model.

metric & model	mean diff	P(T test)	P(wilcoxon te	P(ks test)
ROUGE-2:Baseline ROUGE-2:Fusion	0.113623	0.01044919	0.00659227	0.13997518
ROUGE-2:Baseline ROUGE-2:Ordering	0.00412	0.93451404	0.62148995	0.193041652
ROUGE-2:Baseline ROUGE-2:Ordering+Fusion	0.1236096	0.00510506	0.00445661	0.069092435
ROUGE-2:Fusion ROUGE-2:Ordering	-0.109503	0.01870175	0.00564704	0.069092435
ROUGE-2:Fusion ROUGE-2:Ordering+Fusion	0.0099866	0.79877547	0.75334568	0.999996899
ROUGE-2:Ordering ROUGE-2:Ordering+Fusion	0.1194896	0.00993021	0.0009837	0.031376652
ROUGE-SU4:Baseline ROUGE-SU4:Fusion	0.113623	0.01044919	0.00659227	0.13997518
ROUGE-SU4:Baseline ROUGE-SU4:Ordering	0.00412	0.93451404	0.62148995	0.193041652
ROUGE-SU4:Baseline ROUGE-SU4:Ordering+Fusior	0.1236096	0.00510506	0.00445661	0.069092435
ROUGE-SU4:Fusion ROUGE-SU4:Ordering	-0.109503	0.01870175	0.00564704	0.069092435
ROUGE-SU4:Fusion ROUGE-SU4:Ordering+Fusion	0.0099866	0.79877547	0.75334568	0.999996899
ROUGE-SU4:Ordering ROUGE-SU4:Ordering+Fusior	0.1194896	0.00993021	0.0009837	0.031376652




Mean and P-value of models with ROUGE-2
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
ROUGE-2:Baseline ROUGE-...
ROUGE-2:Baseline ROUGE-...
ROUGE-2:Baseline ROUGE-...
ROUGE-2:Fusion ROUGE-...
ROUGE-2:Fusion ROUGE-...
ROUGE-2:Ordering ROUGE-...
mean diff
P(T test)
P(wilcoxon test)
P(ks test)


Mean and P-value of models with ROUGE-SU4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
ROUGE-SU4:Baseline ROUGE-...
ROUGE-SU4:Baseline ROUGE-...
ROUGE-SU4:Baseline ROUGE-...
ROUGE-SU4:Fusion ROUGE-...
ROUGE-SU4:Fusion ROUGE-...
ROUGE-SU4:Ordering ROUGE-...
mean diff
P(T test)
P(wilcoxon test)
P(ks test)