Before we start working with this toy dataset, I really want to say that I know this dataset before, we can find this dataset from Kaggle or UCI Machine Learning Repository. There should be another csv file called ‘‘trial’’ in the original dataset with 18 columns.

Exploration of Data Analysis

1. The variable title “Score_B” appears 2 times, but their data is not the same
2. The variable “Detection_Risk” only has one kind of variable, so we do not need to take it into account.
3. There are two kind of labels in this dataset. One is “Risk” which is classification label, the other one is “Audit_Risk” which is regression label. We can do these two tasks based on these two labels.

```
#I think this is a very easy classification problem, the label is Risk
#So we can understand it as to predict whether a company is risk or not
from sklearn import tree
import csv
import scipy
from scipy.stats import pearsonr
filename = open('audit_risk.csv')
reader = csv.reader(filename)
audit_risk = [row for row in reader]
name = audit_risk[0]
data = audit_risk[1:]
label = [int(a[len(data[0])-1]) for a in data]
def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass

    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass

    return False
cnt = 0
for a in data:
    for b in a[:len(name)-1]:
        if is_number(b) == False:
            cnt += 1
print(cnt)
#I find that the count of not number is only 4, so I use 0 to replace this
```

```
4
[Finished in 0.5s]
```

4. Threshold to trigger risk: I believe there should be a threshold to classify whether an Audit_Risk is Risk. So I write a program to find the threshold number and find if Audit_Risk is

larger than 1.0144, it will be classified as 1; otherwise it will be classified as 0. To prove this, I write another program to see whether there is an outlier and find there is no outlier.

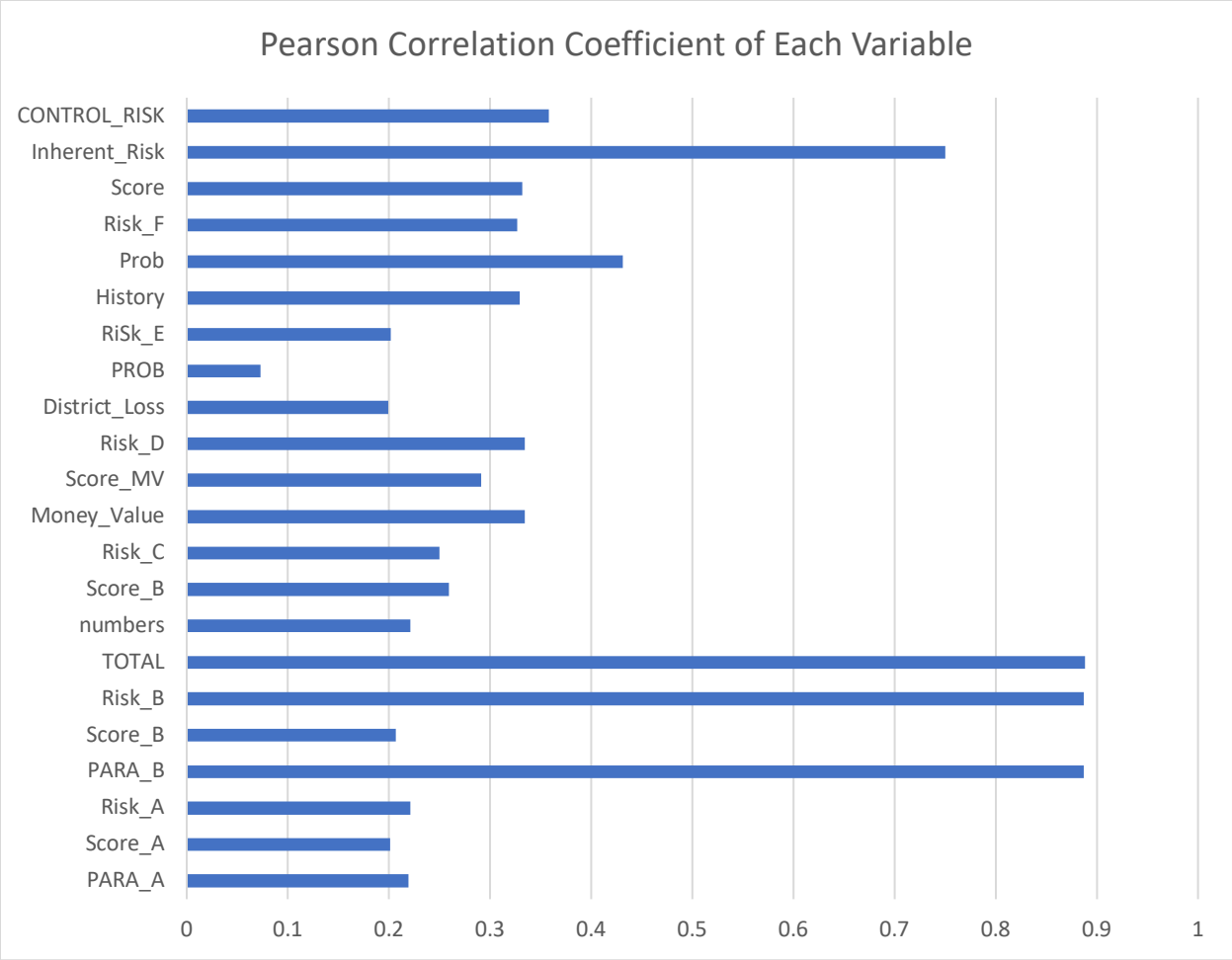
```
#Check the threshold to distinguish whether a company is risk based on Audit_Risk
Audit_Risk = [a[-1] for a in data_only]
threshold = 1.7148
for i in range(len(Audit_Risk)):
    if label[i] == 1:
        temp = Audit_Risk[i]
        if temp < threshold:
            threshold = temp
print(threshold)
outlier = 0.5108
for i in range(len(Audit_Risk)):
    if label[i] == 0:
        temp = Audit_Risk[i]
        if temp > threshold:
            outlier = temp
print(outlier)
```

```
1.0144
0.5108
[Finished in 0.5s]
```

5. Weight of each variable: When we get a new dataset with many columns of variables, one thing we want to know most is which column is the most important one. For the problem of classification, we can use KL divergence or logistic regression to calculate the weight of each variable. For the problem of regression, we can use Pearson correlation coefficient to get the weight of each variable. So I calculate the weight with the later one:

```
correlation_list = []
y = scipy.array(Audit_Risk)
for i in range(2, len(name)-3):
    Inherent_Risk = [a[i] for a in data_only]
    x = scipy.array(Inherent_Risk)
    correlation, p_value = pearsonr(x, y)
    correlation_list.append({name[i]:correlation})
print(correlation_list)
```

```
[{'PARA_A': 0.21975913677122935}, {'Score_A': 0.20184693316925936}, {'Risk_A': 0.2215811172563867},
{'PARA_B': 0.8877941445895132}, {'Score_B': 0.20797542192599608}, {'Risk_B': 0.8875700145287967},
{'TOTAL': 0.8880964563623329}, {'numbers': 0.22146086511866026}, {'Score_B': 0.25973663694912347},
{'Risk_C': 0.25002526392101554}, {'Money_Value': 0.3340832754519391}, {'Score_MV': 0.29172151334071533},
{'Risk_D': 0.3341841482358157}, {'District_Loss': 0.1994257108165942}, {'PROB': 0.07382950565602886},
{'RiSk_E': 0.2029831619366057}, {'History': 0.32977260374735395}, {'Prob': 0.4313099066706606}, {'Risk_F':
0.3277502317339296}, {'Score': 0.3329927559192095}, {'Inherent_Risk': 0.7509030567132486},
{'CONTROL_RISK': 0.35801326948387224}]
[Finished in 0.5s]
```



We can see that the weights of “PARA_B”, “Risk_B”, “TOTAL” and “Inherent_Risk” are very high compared with the others, so these variables are the most important variables.

Data preprocessing

We use decision tree model in sklearn to classify our data. To make it, we have to make sure all of our input in number instead of string. I write a “is_number” function to distinguish whether a string can transform into number. I find that 3 strings are made of English letters and the other 1 is space. So the count of no number is only 4, I use 0 to replace them as follows:

```
#I find that the count of not number is only 4, so I use 0 to replace this
data_only = []
for a in data:
    data_only_element = []
    for b in a[:len(data[0])-1]:
        if is_number(b) == False:
            b =0
        data_only_element.append(float(b))
    data_only.append(data_only_element)
```

Modelling and predict with Decision Tree

The number of the dataset is 776, I use 676 of them as training dataset and 100 of then as testing dataset. I use the Decision Tree model in sklearn as my classification model:

```
70    dtc = tree.DecisionTreeClassifier(criterion="entropy")
71    clf = dtc.fit(data_only[:676], label[:676])
72    result = clf.predict(data_only[676:])
73    correct = 0
74    for i in range(len(result)):
75        if result[i] == label[676:][i]:
76            correct += 1
77    print('Accuracy: {:.2%}'.format(correct/100))
78
```

```
Accuracy: 100.00%
[Finished in 0.5s]
```

At first, you may think this result means overfitting. But considering that our dataset is very small, so I think this result is acceptable.