COMP 309
Assignment four

Name: Siwen Feng
ID: 300363512

Part one

Step one: Load data

The data name is called diamonds.csv. I used pd.read_csv to read this data file. This data file is clean and well organized. Because it doesn't contain any missing values. Features are easy to follow. Hence, the quality of this data is high (low complexity and low noises).

Step two: Initial data analysis

This dataset contains many attributes which are related to diamonds. The purpose of this task is to use these features to identify the price of diamonds. There are 9 features in total. 6 features are numeric and other three are nominal. In the initial observation, we can find the basic characters of this data set. The following statistics revealed some interesting points of this dataset.

	carat	cut	color	clarity	depth	table	x	y	z	price
count	53940.000000	53940	53940	53940	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
unique	NaN	5	7	8	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	Ideal	G	SI1	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	21551	11292	13065	NaN	NaN	NaN	NaN	NaN	NaN
mean	0.797940	NaN	NaN	NaN	61.749405	57.457184	5.731157	5.734526	3.538734	3932.799722
std	0.474011	NaN	NaN	NaN	1.432621	2.234491	1.121761	1.142135	0.705699	3989.439738
min	0.200000	NaN	NaN	NaN	43.000000	43.000000	0.000000	0.000000	0.000000	326.000000
25%	0.400000	NaN	NaN	NaN	61.000000	56.000000	4.710000	4.720000	2.910000	950.000000
50%	0.700000	NaN	NaN	NaN	61.800000	57.000000	5.700000	5.710000	3.530000	2401.000000
75%	1.040000	NaN	NaN	NaN	62.500000	59.000000	6.540000	6.540000	4.040000	5324.250000
max	5.010000	NaN	NaN	NaN	79.000000	95.000000	10.740000	58.900000	31.800000	18823.000000

The chart gives a general idea of how the data been distributed. As we can observe from statistics, distribution of x, y, and z these three attributes having unusual patterns. Min value is 0 and max value is unfairly big. For this scenario, we can deduce those values could highly possible be outliers. In the following process, it is required to get rid of them.

Step three: Data preprocessing

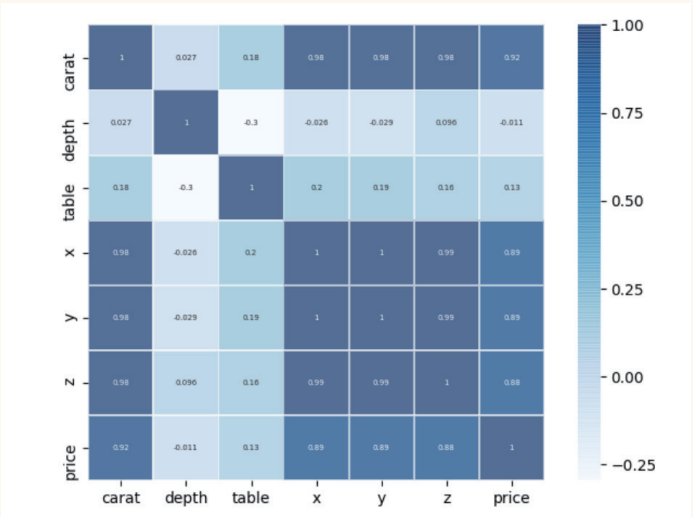
Examining data quality:

This dataset is well organized and has a clear structure. However, from the observation of the dataset, we still can find problems for the dataset which limit us to get better performance.

- Irrelated attributes.
- Outliers.
- Nominal to numbers.

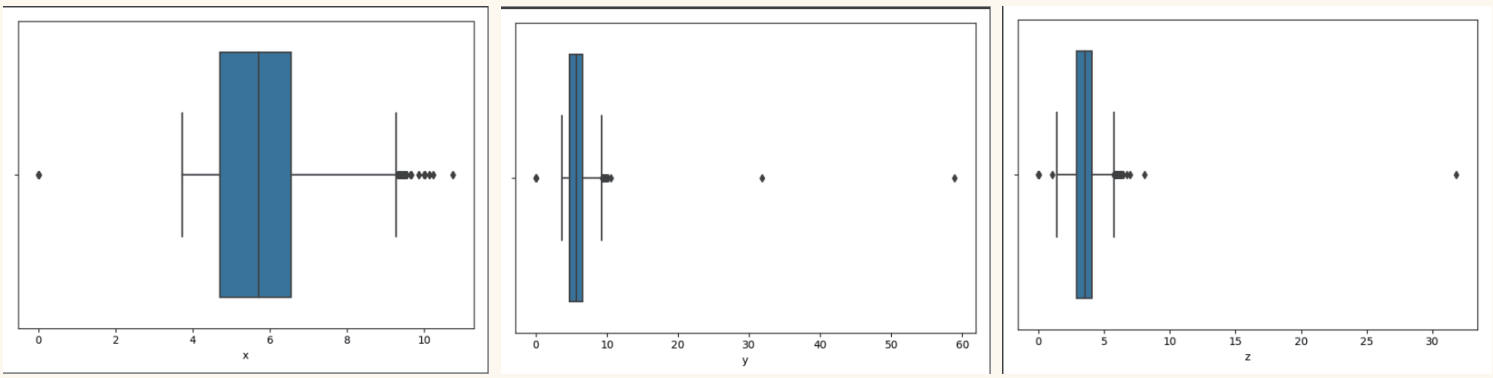
Attributes selection:

This is the correlation heatmap for this dataset. As we can see, depth and table have a very small relationship with price. However, the price is our targeted Y variables. If we put depth and table into our regression process may cause larger errors. Thus, I filter out these two attributes as noises. All algorithms have improvements in their performance after I did attribute selection. Which revealed that the attributes I deleted are noises for the regression training.



Outliers or extreme values:

In the initial observation, we found some unusual values in feature x, y, and z. To ensure that my judgments are correct. I need to give more evidence to support my idea. Hence, I visualized three features individually. they will give us a more clear visualization.



From the figures, we can see, all x, y, and z features have some outliers. Min values are not accurate at all. Based on the box plot above, I deleted some extreme values and make the data in a reasonable range.
So the characteristics of the regression results will not be affected by outliers, so it could provide higher accuracy.

Nominal values to numbers:

The process of regression is to input x variables to get y variables. the relations between x and y could provide a lot of information to us. However, the limitations of this technique are that it only can use numbers to predict. In our data set, there are three nominal features which are not allowed to use as an input. I quantify nominal values into some reasonable numbers to satisfy regression's requirements.

			Cut				
	Ideal	Premium	Very good	Good	fair		
	10	9	8	7	6		
			Clarity				
IF	VVS1	VVS2	VS1	VS2	SI1	SI2	I1
10	9	8	7	6	5	4	3
			Color				
J	I	H	G	F	E	D	
10	9	8	7	6	5	4	

To solve this problem, I follow the principle that better quality has a higher score. 10 will be the highest mark and less good one will have a lower mark. Thus, all nominal values have been changed to numbers and we can use it for our regression. The next part is the code of this conversion.

```
df['cut'] = df['cut'].map({'Ideal':10, 'Premium':9, 'Very Good':8,'Good': 7,'Fair':6})
df['clarity'] = df['clarity'].map({'IF':10,'VVS1':9,'VVS2':8,'VS1':7,'VS2':6,'SI1':5,'SI2':4,'I1':3})
df['color'] =df['color'].map({'J':10, 'I': 9, 'H':8, 'G':7, 'F':6,'E':5,'D':4})
```

Step four: Application of 10 algorithms

10 algorithms are used to predict Y variables. But how we judge their performance? There are five aspects will help us to decide the performance of ten algorithms -- MSE, RMSE, R2, MAE, and execution time. Therefore, we can get performance metrics regarding these algorithms

	Linear Regression	KNN	Ridge regression	Decision tree	Random forest	Gradient Boosting	SGD	SVR	Linear SVR	ML perceptron
R2:	0.91	0.98	0.91	0.96	0.98	0.98	0.91	0.55	0.85	0.96
Ranking:	6	1	6	3	1	1	6	10	9	3
MSE:	1424354.02	385312.36	1424318.6	571413.65	336194.03	383581.45	1440404.03	7365271.62	2425186.82	583200.12
RMSE:	1193.46	620.74	1193.45	755.92	579.82	619.34	1200.17	2713.9	1557.3	763.68
Ranking:	7	3	6	4	1	2	8	10	9	5
MAE:	798.53	310.52	798.54	368.07	290.66	345.65	798.95	1235.11	847.06	433.44
Ranking:	6	2	7	4	1	3	8	10	9	5
Time:	0.01	0.490715	0.004955	0.152066	1.125059	1.575963	0.026288	79.24813	0.06219	23.590549
Ranking:	2	6	1	5	7	8	3	10	4	9

Top three performers are random forest , gredient boosting, and KNN. They provide fewer errors and higher R2. They also have a reasonable execution time. These three algorithms could use as regression predictor to deal with more problems like a diamond. However, on the other side, SVR provides the worst performance and the long execution time (79s). it the worst performer so far.

Step five: Tuning parameters

All results are generated by the defaulted parameters. In this step, I will Implement changes in parameters and investigate the difference brought by.

After tuning the parameters, I found most all of them will have worse performance. However, Three of the algorithm have improvements in their performance. SVR, Linear SVR, and MLP have a significant increase in performance. Thus, I only change the parameters for these three algorithms. The next part displayed where I have changed.

SVR	Kernel = 'rbf' ,C = 100
Linear SVR	C =5.0 ,loss = 'squared_epsilon_insensitive', dual = True
ML perceptron	hidden layer = 100,activation = 'relu' solver = 'lbfgs',learning rate = 'adaptive'

After I alter three algorithms parameters, the performance for them increases dramatically as well. the comparison chart is down below. Red words showed improvements. the green parts mean the drop in the performance.

	SVR	Linear SVR	ML perceptron
R2:	0.55/0.97	0.85/0.91	0.96/0.98
MSE:	7365271.62/453457.02	2425186.82/1426739.2	583200.12/347657.16
RMSE:	2713.9/673.39	1557.3/1194.46	763.68/589.62
MAE:	1235.11/336.45	847.06/799.65	433.44/332.4
Time:	79.24813/71.559224	0.06219/5.629415	23.590549/16.113674

After deployed new parameters for SVR, it provided a much better result than defaulted parameters, which reveals that SVR also can be an effective algorithm if we deploy suitable parameters. It still consumes a long time to execute though. It is still a good try which inspired us to try different parameters and test what the influence brought by these parameters.

Step six: updated rankings

	Linear Regression	KNN	Ridge regression	Decision tree	Random forest	Gradient Boosting	SGD	SVR	Linear SVR	ML perceptron
R2:	0.91	0.98	0.91	0.96	0.98	0.98	0.91	0.97	0.91	0.98
Ranking:	7	1	7	6	1	1	7	5	7	1
MSE:	1424354.02	385312.36	1424318.6	571413.65	336194.03	383581.45	1440404.03	453457.02	1426739.2	347657.16
RMSE:	1193.46	620.74	1193.45	755.92	579.82	619.34	1200.17	673.39	1194.46	589.62
Ranking:	8	4	7	6	1	3	10	5	9	2
MAE:	798.53	310.52	798.54	368.07	290.66	345.65	798.95	336.45	799.65	332.4
Ranking:	7	2	8	6	1	5	9	3	10	4
Time:	0.01	0.490715	0.004955	0.152066	1.125059	1.575963	0.026288	71.559224	5.629415	16.113674
Ranking:	2	5	1	4	6	7	3	10	8	9

Red - rank up Green - rank down

Part two

Step one: Load data

In this part, we need to do the classification task. We have received two data sets, one is a training set and one is testing set. these two datasets contain census information related to a person. For this assignment, I load two datasets separately by using pd.read_csv. However, these two sets have a slight difference which I will give a detailed explanation in the following steps

Step two: Data understanding

adults data doesn't contain labels. Thus, the first step I should find out what meaning every feature represent for. Based on the given information, we can match features with their labels.

```
["Age", "Workclass", "Final_Weight", "Education", "Highest_Grade", "Marital_Status", "Occupation",
    "Relationship", "Race", "Sex", "Capital_Gain", "Capital_Loss", "Hours_per_Week", "Native_Country", "Income_Class"]
```

After understanding the data, we should consider what we should modify for the classification process. preprocessing the data set could help us have a deep understanding of the dataset.

Step three: Data preprocessing

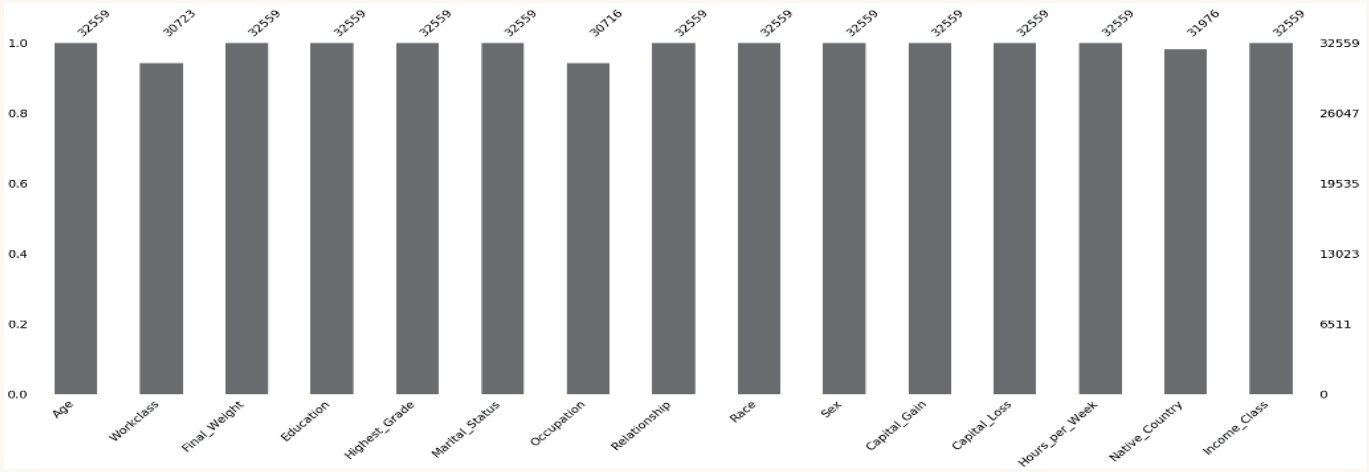
Examining data quality:

This dataset is not an ideal dataset for us to do classification. The complexity of the data set is high and the noise is also high as well. both indicated that the quality of the dataset is low. For optimizing dataset, I should take some measurements.

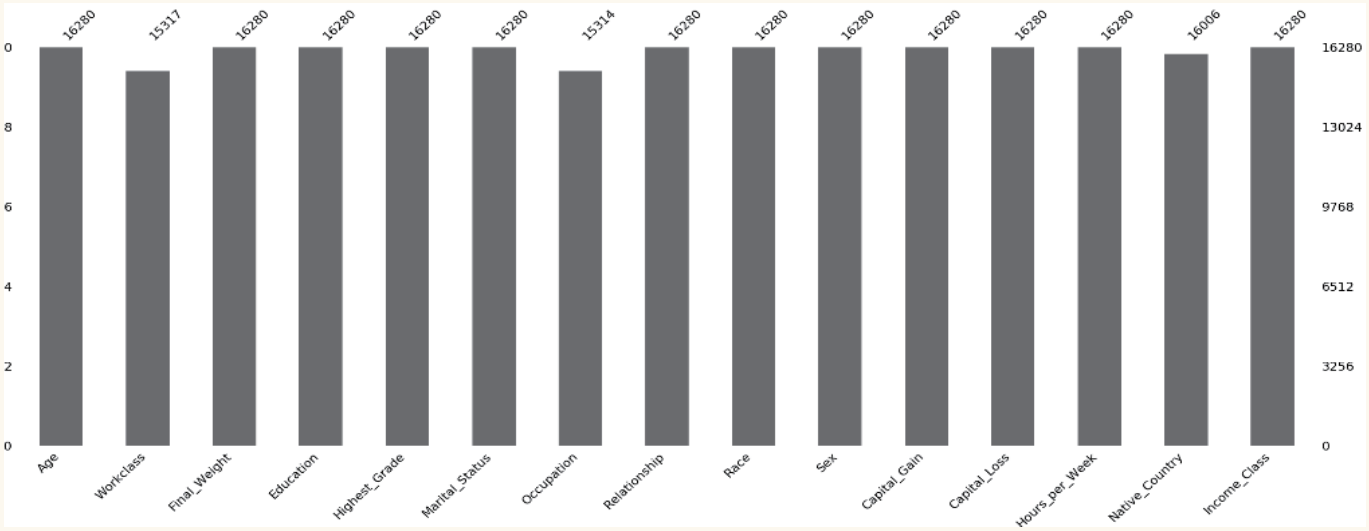
- Replace missing values
- Attributes selection
- Change classifier to binary style
- Match test and training set

Replace missing values:

To detect missing values, there are multiple method could make it. I used missingno package to help me draw all features plots. The results are shown below.



Training data



Testing data

Both training and testing sets have missing values in work class, occupation, and native country. Because the type of data is the category. It is not easy to find a specific value to replace them. The safest method to get rid of them is to replace them with the null values. I used the 'replace' function to complete this step.

```
adult_data = adult_data.replace(' ?', np.nan)
adult_test = adult_test.replace(' ?', np.nan)
```

Attributes selection:

Almost all features in adult's data are categorical attributes which means we are impossible to find the correlation for our targeted feature. Instead of correlation, I could only investigate the relationship of each feature to judge whether it is noise or not. Therefore, there are two attributes I think will bring a negative impact on my final result.

Education and highest grade represents the same meaning, However, education is a categorical item which eventually should convert into the number to do classification. However, the highest grade is numeric education. We don't have to keep both of them. Thus, in this scenario, I delete eduction make data more clear.

Bachelors	13
Bachelors	13
HS-grad	9
11th	7
Bachelors	13
Masters	14
9th	5
HS-grad	9
Masters	14
Bachelors	13
Some-colle	10
Bachelors	13

The second feature needs to filter out is the final weight. Final weight represents estimated populations who have similar background or census information. However, It could be a big noise for our classification target. The target of this assignment is to predict income classes. However, Final weight is a very specific number have little relationship with income. Thus, it makes the prediction have lower accuracy. In attribute selection, I deleted this column for simplifying data.

77516
83311
215646
234721
338409
284582
160187
209642

I used the 'drop' function to delete these two features. The results have been improved after I did attributes selection.

```
df.drop('Final_Weight', axis=1, inplace=True)
df.drop('Education', axis=1, inplace=True)
```

Change income class into binary style:

The eventual goal of this task is to predict a person's income level. In our label, we only need to find out two types of income classes (>50k and <50K). Change label into 0 and 1 is easier to manipulate for a later step.

```
y_adult_data = y_adult_data.replace(" >50K", 1)
y_adult_data = y_adult_data.replace(" <=50K", 0)
y_adult_test = y_adult_test.replace(" >50K.", 1)
y_adult_test = y_adult_test.replace(" <=50K.", 0)
```

Match testing and training set

The last step of preprocessing is to validate they share same columns. When I used one hot coding to convert all features into numbers. I find that they have a different number of columns.

Because there is a country only appeared in the training set. The country name called " Holand-Netherlands". It caused inconsistent columns between testing and training set. Thus, I deleted Holand-Netherlands this columns. Then, testing and training set are matching with each other now.

```
['Age', 'Workclass'
(32560, 88)
['Age', 'Workclass'
(16280, 87)
```

```
adult_data = adult_data[~(adult_data.Native_Country == ' Holand-Netherlands')]
```

Step four: Application of 10 algorithms

In this step, I will implement 10 classification algorithms and find out their accuracy, precision, recall, f1 score, and AUC. Thus, these few elements could help me to build a complete understanding regarding their performance.The metrics of 10 algorithms will be shown below.

	KNN	Naïve Bayes	SVM	Decision tree	Random forest	Ada Boost	Gradient Boosting	Linear discriminant	ML perceptron	Logistic regression
Accuracy:	0.85	0.7922	0.8684	0.8181	0.842	0.8616	0.8708	0.8454	0.8476	0.8536
Ranking:	5	10	2	9	8	3	1	7	6	4
Precision:	0.6955	0.5396	0.8224	0.6182	0.6975	0.7595	0.7963	0.7108	0.682	0.7319
Ranking:	7	10	1	9	6	3	2	5	8	4
Recall:	0.6492	0.8203	0.5647	0.6019	0.585	0.6061	0.6089	0.5827	0.6648	0.5998
Ranking:	3	1	10	6	7	5	4	8	2	9
F1_Score:	0.6716	0.651	0.6696	0.6099	0.6363	0.6742	0.6901	0.6404	0.6733	0.6593
Ranking:	4	7	5	10	9	2	1	8	3	6
AUC:	0.7807	0.8019	0.7635	0.7435	0.7533	0.7734	0.7804	0.7547	0.7845	0.7659
Ranking:	3	1	7	10	9	5	4	8	2	6

Is accuracy the best performance metric to evaluate a classifier? And why?

Accuracy represents the ratio of correctly predicted instances. It is a good aspect to judge the performance of the algorithm. In many cases, it is a useful indicator. However, accuracy is very dependent on the quality of the data. If the data have outliers, extreme values, or imbalanced class distribution, the accuracy will drop significantly.

Unfortunately, the training set is an imbalanced data set which means the model we predict maybe not very accurate. The training data have only around 25% instance belong to the class '< 50k'. It makes sense in reality because only a minority of people in the whole population belong to the wealthy class. The data set is not fairly distributed instances. The result we get also reveals the consequence -- all 10 algorithms don't have good accuracy. There is no accuracy bigger than 90%.

In this case, AUC, precision, and recall all can take account into to evaluate the performance. AUC is good at dealing with binary classification problems. Precision-Recall could prevent inaccurate high performance when the negative numbers have a larger portion in the dataset.

Find the two best algorithms according to each of the four performance metrics. Are they the same? Explain why.

The best performer for this assignment is gradient boosting. Adaptive boosting is another algorithm which is similar to gradient boosting classifier. From the name, we can tell they are belonging to the same algorithm family - boosting algorithm. However, they still have their characters distinguith with each other.

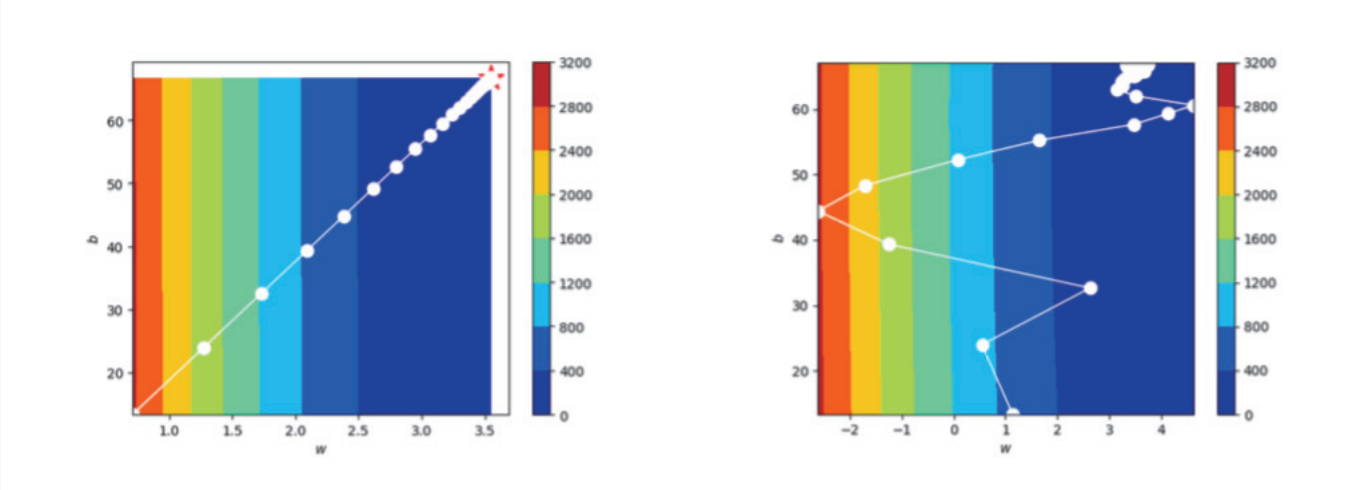
During the process of classifing, both boost a group of weak features into a strong pattern. The difference between them is how two algorithms generate weak features.

Adaptive boosting alter the sample distribution and change the weight of each instances. When weight will be various. the weights will increase when it is wrongly predicted instance and vice versa. After the model building process. the weaker leanrers will be added to strong learner and contributes to its performance.

Gradient boosting doesn't change sample distribution at all but it used gradient descent to optimise the performance. the performace get improved by calculating the gredients. The weaker learner trained from the existing errors generated by strong learner. Hence, they have different inner methods to deal with weaker learner.

Part three

a) Gradient Descent paths of BGD+MSE and MiniBatchBGD+MSE



BGD will upgrade the model by calculating the errors in the training set to minimizing the loss function. it will update the model after finish one cycle. Thus, it shows a stable improvement. After a few times of execution, the number of errors will decrease and the model will reach an optimal point. The graph shows a linear trend. However, if the size of the training set is large, the execution cost will significantly rise.

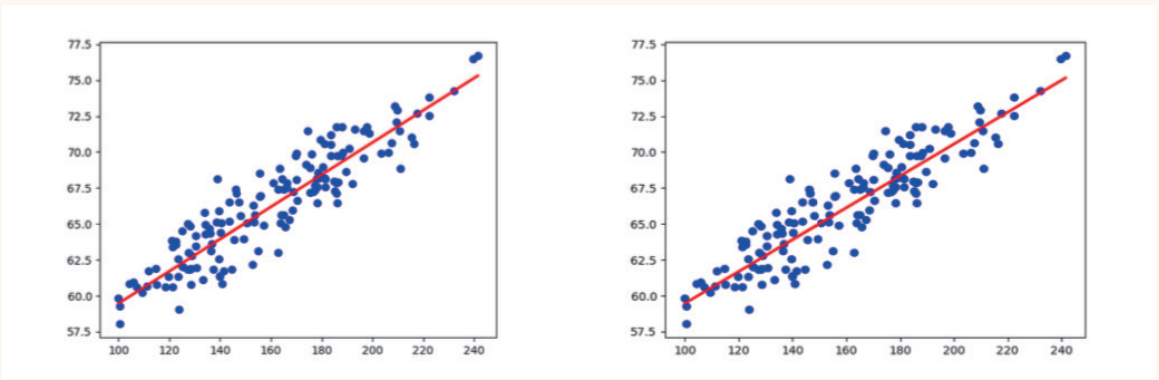
The path of MBGD is fluctuating and didn't show a linear trend. One of the reason is that it splits data into smalls size data sets and then using small batches to find errors. It speeded up the whole process and leads to a fluctuating path. MBGD takes advantage of SGD and overcomes some drawbacks of both SGD and BGD. It can consider as an optimized algorithm of them.

b) MSE, R-Squared, and MAE for four models

	MSE	R2	MAE
MSE + BGD	2.41	0.84	1.28
MSE + MBGD	2.44	0.84	1.28
MSE + PSO	2.41	0.84	1.28
MAE + PSO	2.43	0.84	1.28

As we can see, the R2 and MAE numbers are the same which means four models have very small difference in this scenario. The only different part is the MSE. BGD + MSE and PSO + MSE have same result and ranking top two. But the variation amongst fout theniques still very small in the range (0.01~0.03).

c) Generate a scatter plot with the regression line learnt by PSO+MSE and PSO+MAE and the data points

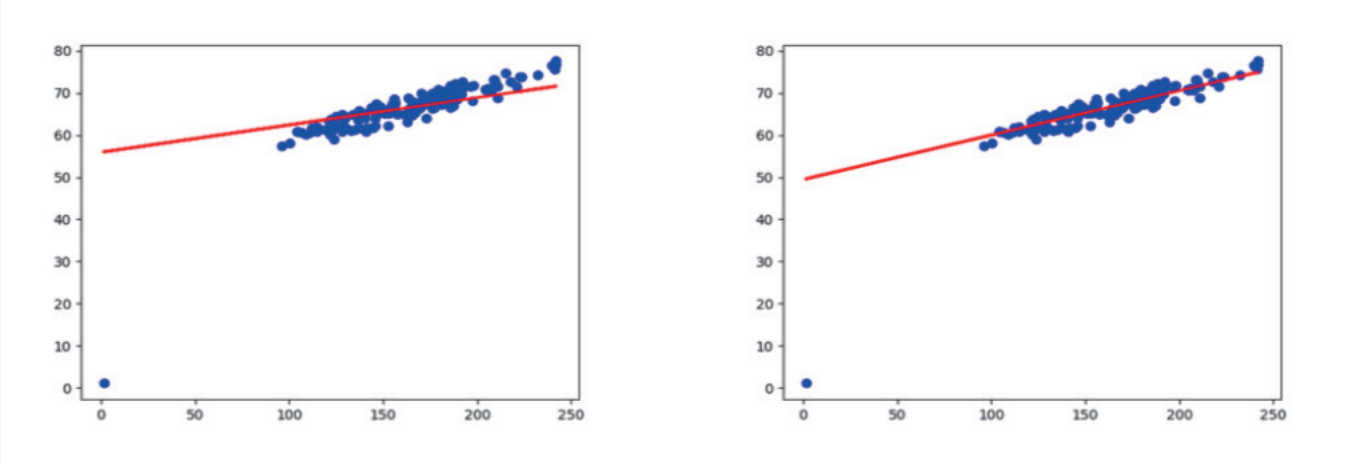


d) Compare the computational time of the BGD+MSE, MiniBatchBGD+MSE and PSO+MSE methods,find out the fastest one and slowest one, and explain why.

	Execution time
MSE + BGD	0.005
MSE + MBGD	0.112
MSE + PSO	0.582
MAE + PSO	0.707

BGD + MSE have least execution time and PSO + MAE have longest exectution time. The reason also easy to understand. Like we analyzed diagram in a). When BGD deal with small data set they can quick run training set and find errors. More BGD runs, the less errors we can get. At the end stage, the algothimen react very fast becuse there no more errors to detect. the dataset is small in out case. Therefore, BGD is fastest. Both PSO runing longer time because they are involved with high complex computual calculations, then the speed naturally slower than BGD and MBGD.

a) Scatter plots with regression line learnt by PSO+MSE and PSO+MAE in test set



b) Compare the above two plots with the two plots you draw in Question 1(c), and discuss which of the two methods (PSO+MSE or PSO+MAE) is less sensitive to outliers and explain why

The previous two diagrams are entirely identical, it reveals that both algorithms have high similarities. Even we take account into outliers. The result was still similar. Based on the diagram, we can find MSE is more sensitive than MAE. PSO + MAE still fit the regression line and PSO + MSE shows a small slope. From the given data, we can find the regression equation drops as well.

MSE measures the distances between the scatters and regression line. The outliers are extreme values that will make data set more scatter, Thus, MSE is sensitive with them. MAE will use the absolute values of the dataset. Therefore, MAE can handle the errors brought by outliers well and not enlarge the errors.

c) Discuss whether we can use gradient descent or mini-batch gradient descent to optimise MAE? and explain why.

The cost function of MAE is linear which may lead to a fixed adjustment for each generation. Thus, overshoots may occur and a small or changing learning rate cannot help solve this problem either. Even we keep all other condition same, MSE will provide a better and precise result. In the meantime , when perfect prediction occurs, MAE will lead to uncertain gradient which means gradient descent is not avaiable. Hence, MAE is not an ideal method to use for this scenario.