Name: Dylan Thomson ID: 300419012

Comp309 Assignment 4

Part 1:

Model Parameters:

When reading the documentation I found that the default parameters were quite good, as I am sure whoever wrote the algorithms is quite clever. The parameters I did use were ‘early stop’ and ‘max iterations’ for SVR and MLP. I used these during development to make the execution time shorter allowing me to rapidly test. However, once I was certain my pipeline was complete I removed these parameters allowing the models to train for the full duration.

Preprocessing:

My preprocessing of the data included converting the nominal attributes into numeric values. Sklearn requires only numeric values. I converted these attributes using a label encoder. Clarity and Cut are ordinal attributes so I have labelled them using this ordering. Color is a categorical attribute, I considered using a one hot encoder to process this attribute which would split it into k binary columns represented the k categories. I decided against this in the end as it would add a lot more attributes to the dataset increasing its execution time.

Once all my attributes were numeric I standardized the values. This converts the values into a normal distribution around 0. Doing so makes it so that attributes with greater magnitudes do not hold more weight in the regression model.

The dataset had no missing values so I did not need to worry about that.

Results:

Model	**Execution Time (s)**	**MSE**	**RMSE**	**R-Squared**	**MAE**
Linear Regression	0.01	1851854.36	1360.83	0.87	855.67
kNN	0.46	665257.85	815.63	0.96	407.81
Ridge	0.01	1851846.11	1360.83	0.87	855.77
Decision Tree	0.15	564018.54	751.01	0.96	365.25

Random Forest	0.93	332782.39	576.87	0.98	284.57
Gradient Boosting	0.97	476371.34	690.20	0.97	374.81
SGD	0.01	1868760.44	1367.03	0.86	872.44
SVM	64.48	8061589.19	2839.29	-1.38	1384.51
Linear SVM	0.02	2759423.79	1661.15	0.70	907.72
MLP	26.48	1143605.40	1069.39	0.92	567.75

Analysis and Conclusion:

By analysing these metrics we gain more insight into the strengths and weaknesses of each model more so than if we were to just use one.
Mean absolute error is a very basic metric which shows the average magnitude that the model is incorrect by. The best performers in the metric are the random forest and the decision tree.

RMSE and MSE are effectively the same giving us values proportionate to the mean squared error. Because these values are averaged after being squared then finally taking the root of the value, errors with a higher magnitude have a larger impact than errors of a small magnitude. This makes these metrics good for identifying models that produce less outliers. In this case the models with the best RMSE are random forest and gradient boosting.

R-Squared as a metric that measure the models “goodness of fit” it measures each points distance to the predicted value giving a r-squared value from [0,1] . Unsurprisingly random forest and gradient boosting have performed very well in this metric. What is more interesting is that SVM obtained a r squared value of -1.38. This is very strange and shows there is something very wrong with how this model was trained (as do the other metrics). A possible explanation is that to get a negative value the model must be worse fit than the null hypothesis (a flat line/constant value).

Finally we have execution time which does not measure how well the models predict values, but rather how long they took. I measured this as the time taken to train the model and predict the test set. Most models were very quick apart from SVM and MLP. MLP took a long time as during training it never converged and continued training until the iteration limit of 200 was hit.
SVM took a long time as its fit time is O(n^2) making it very slow to fit against dataset with over 10000 instances. Our test set had 37759 instances so its long execution time was expected.

Name: Dylan Thomson ID: 300419012

By analysing the information from these metrics I would conclude that the random forest model was definitely the best for this given context. It had the best score in all the metrics except execution time where its value was negligible.

Part 2:

Preprocessing:

The adult dataset contains 10 nominal attributes, from this ‘education’ could be considered ordinal however I see that as an external assumption that could inorganically alter the model so I will consider it as categorical.

Once again I had to convert these nominal attributes into numeric data using the label encoder.

This time the dataset did have missing values. I left these in the dataset as the label encoder would label them as their own category, this seemed logical to me as there may be a correlation of why values may be missing in each attribute.

Afterwards I normalised all these values to be between [0,1], this ensures no attribute has more initial weight in the model. I had to normalise this rather than standardise it because the naive bayes model doesn’t accept negative values in its data.

Results:

In a table here are the results of the ten classification models against five metrics.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**
kNN	0.82	0.56	0.65	0.60	0.73
NaiveBayes	0.77	0.02	0.99	0.04	0.51
SVM	0.83	0.44	0.75	0.55	0.70
Decision Tree	0.81	0.61	0.59	0.60	0.74
Random Forest	0.85	0.58	0.73	0.65	0.76
Ada Boost	0.86	0.59	0.76	0.66	0.77
Gradient boosting	0.87	0.59	0.79	0.68	0.77
Linear discriminant analysis	0.82	0.41	0.69	0.51	0.67

Name: Dylan Thomson ID: 300419012

MLP	0.85	0.59	0.73	0.65	0.76
Logistic Regression	0.82	0.44	0.70	0.54	0.69

Is accuracy the best performance evaluator?

Accuracy is the simplest, most intuitive performance evaluator for classification but is it the best? An issue with accuracy can be seen when the datasets classes are imbalanced. If the classes are 9:1 always picking the most frequent class would give us an accuracy of 0.9 which seems very good, however this model is terrible at classifying the other class. So I would conclude that accuracy can be a good performance evaluator but is not always the best.
This is relevant for the adult test set as it is quite imbalanced as can be seen here.
Test set sample classes:

There are 24720 samples of <= 50K and only 7841 >50K. If our models only classified as <=50K they would obtain an undeserving accuracy of 0.76

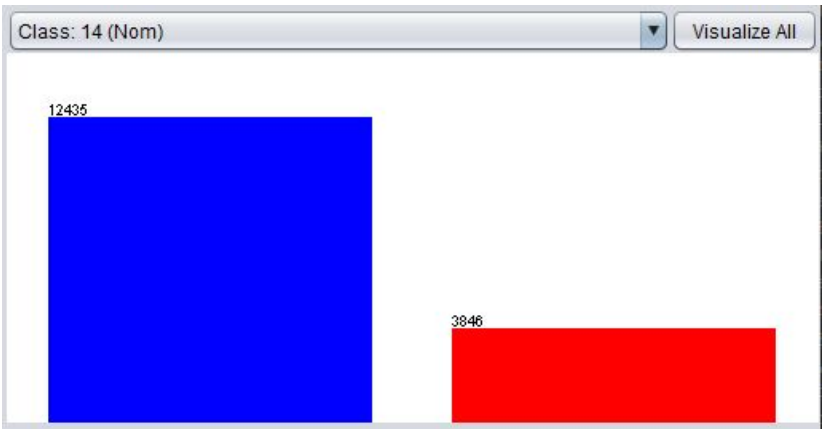
To improve accuracy some changes could be made to its calculation. A balanced accuracy metric could be described by (TruePositiveRatio / TrueNegativeRatio)/2. This equally weights its performance on both classes to give a balanced evaluation. For example this new balanced accuracy metric would give 0.5 for my previously described 9:1 scenario. This is a much better evaluation of the model.

Two best algorithms for each performance metric analysis

Accuracy: Gradient Boosting, Ada boost
Precision: Decision Tree, [ada boost, gradient boosting, MLP]
Recall: Naive bayes, gradient boosting
F1-score: Gradient boosting, ada boosting
AUC: Gradient boosting, ada boosting

Name: Dylan Thomson ID: 300419012

Comparing high achievers for each metric we can see that gradient boosting and ada boost commonly occur. This is a pretty clear indicator that they are the optimal models for this dataset configuration. They have performed well in all aspects and as such I am confident they are the best fit.
Some other notable models are Naive Bayes and Decision tree.
Naive Bayes is the highest rated model using the recall metric, this shows that naive was quite good at predicting true positives, in this case being a salary <= 50K. The fact that it did not perform well in other metrics shows that it was not very good at classifying the other class. With an understanding of how naive bayes works this makes a lot of sense. Since the dataset is unbalanced, with more <=50K samples in the training set, the naive bayes model statistically prefers that class.
Training set sample classes:



Another anomaly is decision tree obtaining the highest precision value. Precision is how often your model correctly identifies positives vs false positives (true positives / (true positives + false positives)). This shows that the decision tree identified >50K's as <=50K's the least. Precision is a good metric to use if identifying false positives could be catastrophic.

Part 3:

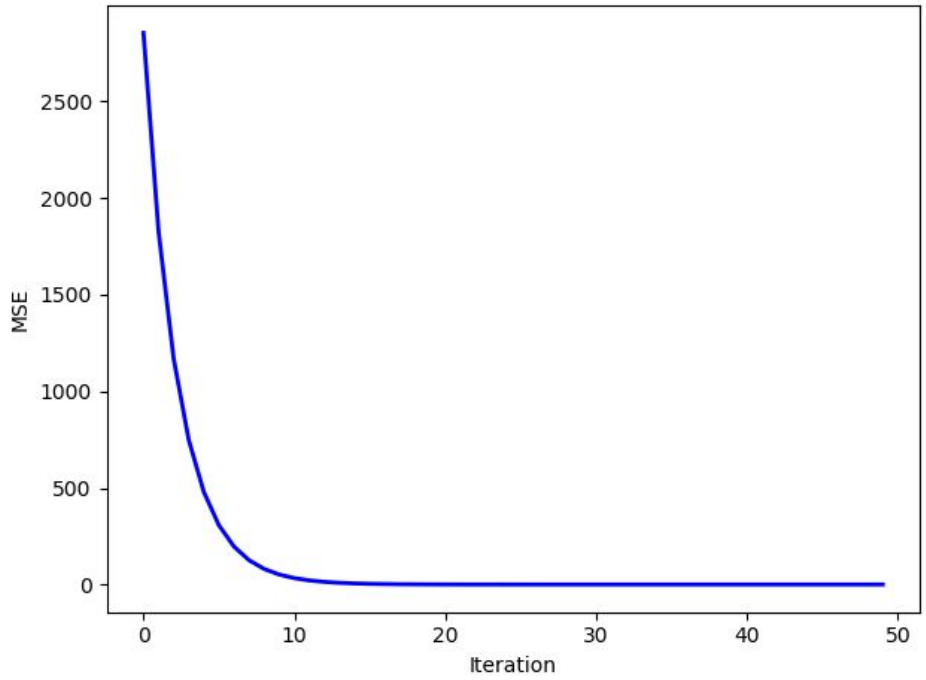
part2.csv

This is a regression task where we use an input height to predict weight. I will be using four methods of optimisation to analyse and visualise their results and behaviour on the part2.csv dataset. This is a dataset that doesn't contain outliers.

Gradient descent paths:

BGD+MSE:

Name: Dylan Thomson ID: 300419012



MiniBGD+MSE:
For miniBGD my charts would not render. Whenever the code reached the point to visualise the training it froze.

Results:

BGD+MSE:
Learn: execution time=0.004 seconds
R2: 0.8362099758134547
MSE: 2.4156981721089568
RMSE: 1.5542516437530174
MAE: 1.280180637895127

Mini BGD+MSE:
Learn: execution time=0.136 seconds
R2: 0.8376130004731006
MSE: 2.395005311706969
RMSE: 1.5475804701878895
MAE: 1.275971811684771

PSO+MSE:
Learn: execution time=0.632 seconds
R2: 0.8365841716337673
MSE: 2.4101792513832936
RMSE: 1.552475201535694

Name: Dylan Thomson ID: 300419012

MAE: 1.2790653151005467

PSO+MAE:
Learn: execution time=0.390 seconds
R2: 0.8356224436697872
MSE: 2.4243635370025873
RMSE: 1.557036780876607
MAE: 1.2831383811067376

Result Analysis:

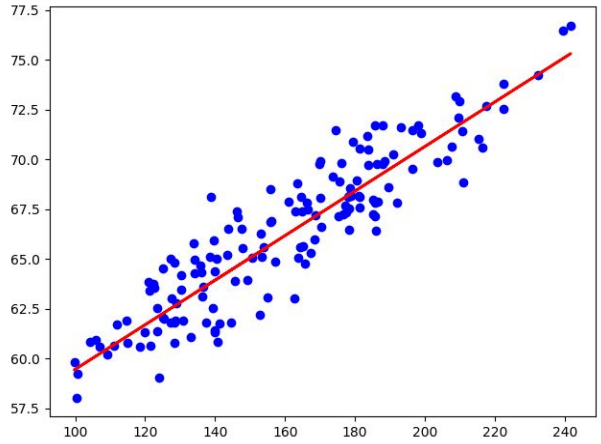
From the results we can see that the model performs very similarly regardless of the optimisation method. There are only slight variations in the metrics values.

Looking into the small variations we can see that miniBGD performed better than BGD at minimising MSE. This is because there is more constant feedback for the model to adjust by making it perform better.

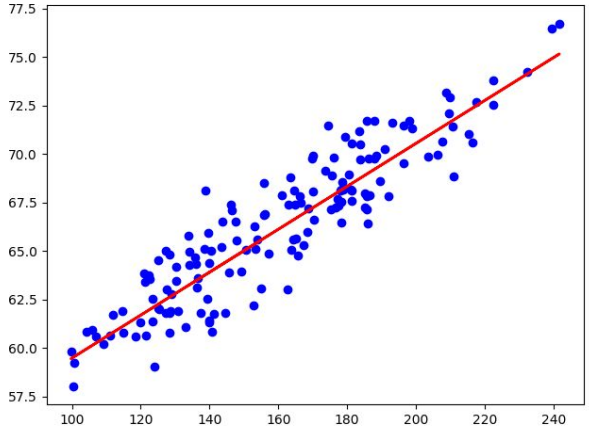
One curious aspect of the PSO optimisers is that when using MAE as its metric it produced a higher MAE value than when MSE was its metric. This is very strange and possibly just coincidence, but interesting nonetheless.

Models vs test data:

PSO+MSE



PSO+MAE



These two graphs are almost identical with PSO+MAE's gradient being slightly shallower. This is likely because without large outliers MSE and MAE are very similar metrics.

Name: Dylan Thomson ID: 300419012

MSE computation times:

BGD: 0.004 seconds
MiniBGD: 0.136 seconds
PSO: 0.632 seconds

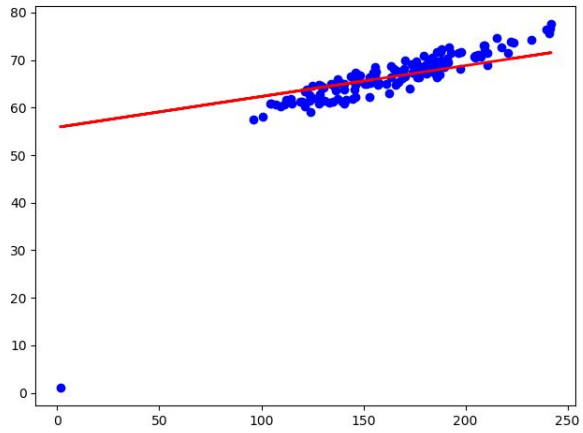
All the methods were optimising in reference to MSE. Comparing BGD and MiniBGD it is obvious that miniBGD would take longer. This is the case because miniBGD does many extra loops within each iteration where BGD only does one pass over the data each iteration.
As for PSO, this method is more taxing so it took longer than the gradient descent methods. Although these times vary for this dataset the difference is negligible and can be mostly disregarded, however, with larger datasets or datasets with more dimensionality this time could increase greatly.

Part2outliers.csv

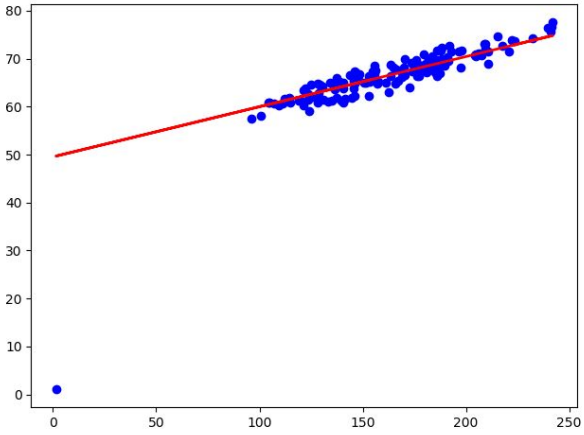
This section is tested using the Part2Outliers.csv dataset, this dataset contains 2 outliers.

Model scatterplots with test data:

PSO+MSE



PSO+MAE



This time the models line's difference is much more noticeable. We can see that when optimised with MSE the line is a worse fit for most of the data. This is because MSE is more sensitive to outliers than MAE. MSE is more sensitive due to the values being squared before they are averaged, so larger errors are more significant.

Can gradient descent use MAE?

Gradient descent methods are unable to use MAE as their metric because MAE is not differentiable. Because it's not differentiable the gradient to follow cannot be found.