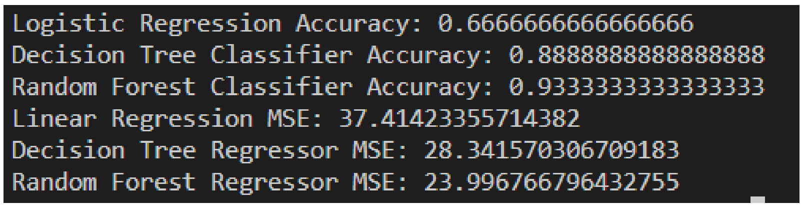
郭雅美 B09902085 Foundations of Artificial Intelligence
Homework 3 – Programming report

a) Compare the performance of the linear, nonlinear, and random forest models on the classification and regression tasks. Based on the evaluation metrics (accuracy for classification and mean squared error for regression), which model performs better for each task, and why do you think this is the case?



```
Logistic Regression Accuracy: 0.6666666666666666
Decision Tree Classifier Accuracy: 0.8888888888888888
Random Forest Classifier Accuracy: 0.9333333333333333
Linear Regression MSE: 37.41423355714382
Decision Tree Regressor MSE: 28.341570306709183
Random Forest Regressor MSE: 23.996766796432755
```

Based on my implementation, Random Forest model performs better for both classification and regression.

Random forest performs better than decision tree for both classification and regression because it generates multiple random forest through the use of random dataset subset. This allows random forest to not experience overfitting and will do better generalization with new dataset.

Random forest has higher accuracy than logistic regression and lower MSE when compared to linear regression because the dataset used for training and testing don't exactly have linear relationship between features and output. Since random forest can have a lot of hyperplanes, random forest can help classify and predict dataset that has complicated and complex relationship between features and output.

b) Apply both normalization and standardization techniques to the datasets and compare their impact on the performance of one of the models (e.g., logistic regression). Explain the rationale behind each technique, and discuss their advantages and disadvantages in the context of the given tasks.

I tried it on linear regression
For linear regression:
- With normalization -> MSE = 37.41
- With standardization -> MSE = 22.41

Normalization will try to make the features down to a range between 0 and 1, while maintaining its proportionality (by doing X-min/ max- min, where X is the features). As a result, it will still maintain the importance/impact of each feature from the original dataset (standardization will lose this since it divides it by standard deviation). Furthermore, standardization take more compiling time since it requires counting the standard deviation.

On the other hand, standardization will try to center the dataset around 0 (it does this by subtracting the mean) and afterward, it will apply scaling (by dividing it with standard deviation). Since it uses mean and standard deviation, the scaling won't be as imbalanced as normalization. As a result, the linear regression model will be less sensitive to outliers. If there are many outliers, the MSE of standardization will most likely to be less than the MSE of normalization.

Therefore, normalization and standardization have their own advantages and disadvantages.

c) Train your logistic or linear regression model using two different configurations of learning rate and number of iterations. Compare the performance of these configurations using the evaluation metrics (accuracy for classification and mean squared error for regression). Discuss how the choice of learning rate and the number of iterations affect the convergence and overall performance of the models.

Here is a table summarizing how the learning rate and iterations affect logistic and linear regression:

			Logistic Regression	Linear Regression
Rows	Learning Rate	Iterations	Accuracy	MSE
1	0.001	100	0.62	186.33
2	0.001	1000	0.62	64.53
3	0.001	10000	0.66	37.41
4	0.01	100	0.62	64.47
5	0.01	1000	0.66	37.41
6	0.01	10000	0.91	60.76
7	0.1	100	0.66	37.41
8	0.1	1000	0.91	60.76
9	0.1	10000	1	63.37

From this table, we can see that for logistic regression, as the number of iterations increases, the more accurate it becomes. This means that the gradient descent is making the error function closer to 0. For logistic, as the learning rate increases, the accuracy gets better. It even reaches an accuracy of 1 when the learning rate is 0.1 and the number of iterations is 10000.

For linear regression, increasing the number of learning rate or the iteration does not mean a better result.

- This can be seen just by seeing the MSE when the learning rate is 0.1 (row 7-9). The MSE for linear regression increases as the number of iterations increases. Overfitting is happening in this case.

- On the other hand, if the learning rate is 0.001, MSE gets better when the number of iterations increase. This mean the gradient descent is indeed making the error function closer to 0.
- Another scenario that can be seen happening is overshooting the optimal solution (the model diverges instead of converging into the optimal solution). This can be seen by comparing row 5 and row 8. MSE got worse when the learning rate increases.

d) Discuss the impact of hyperparameters, such as the number of trees and maximum depth, on the performance of the random forest model for both classification and regression tasks. How do these hyperparameters affect the model's complexity, generalization, and potential for overfitting? Include a brief explanation of how you selected or tuned these hyperparameters in your implementation.

I pick some parameters here and create a table for it:

			Random Forest Classifier		Random Forest Regressor	
Rows	No of trees	Max Depth	Time (sec)	Accuracy	Time (sec)	MSE
1	100	5	0.87	0.93	27.96	25.06
2	100	10	0.88	0.91	50.26	23.53
3	100	20	0.90	0.93	55.45	23.22
4	1000	5	8.83	0.93	272.55	23.58
5	1000	10	8.65	0.93	495.40	23.41
6	1000	20	8.62	0.93	570.08	23.45
7	5000	5	43.93	0.93	--	--
8	5000	10	44.13	0.93	--	--
9	5000	20	44.10	0.91	--	--

note: I don't really do 2000 trees for random forest regressor because it takes way too much time to run.

For random forest classifier:

- the accuracy does not differ by much when I tried to change around the number of trees or the max depth parameter.
- However, there is overfitting occurring in row 2 and row 9, but nonetheless, it only differs by 0.02. The overfitting might also be because that it keeps on selecting the same subset of dataset.
- Increasing the number of trees is not worth it since the accuracy stays quite consistent through out the test, but adds a lot of time and increase the model complexity.
- Changing the tree depth does not really contribute to time complexity.

Overall, it seems that out of all the possible number of trees and max depth in the table, the best one is row 1, with a parameter where the number of trees is 100, and the max depth is 5.

For random forest regressor:

- When the number of trees is fixed on 100, the MSE decreases when the max depth increases. From max depth of 5 to 10, the MSE decreases by 2. However, through the rest of the parameters, it seems that the MSE stay quite consistent with a value of 23. Thus, I don't think it is worth it to increase the number of trees or the max depth even more, since increasing those parameters will increase the compiling time.
- Overfitting occurred when increasing the tree depth from 10 to 20 (row 5 to row 6), increasing the MSE by 0.04.

Overall, it seems that for random forest regressor, the parameter of 100 number of trees and 20 max depth seems to be the best.

When tuning the hyperparameters, I like to experiment around with the number of trees and max depth (I do a distribution of ranges), and pick the one that has the right balance of accuracy/MSE along with compiling time.

e) Analyze the strengths and weaknesses of the implemented models. In what scenarios would you choose to use a linear model over a nonlinear model, or a random forest model, and vice versa? Discuss the trade-offs between model complexity, interpretability, and performance.

<u>Linear Regression / Logistic Regression:</u>

Strength

- Good for features and output that have a linear relationship.
- For linear regression, it is not necessarily need to implement gradient descent. Can directly use the equation where the derivative of error function is 0 to find the optimal weight.
- As a result, does not take much compile time.
- Easier to implement (compared to decision tree and random forest)

Weakness

- For multiclass logistic regression, need to do further one hot encoding.
- Bad at complex relationship.
- Always assume that features are independent of each other.

<u>Decision Tree:</u>

Strength

- Can handle complex relationship between features and output.
- Handle large dataset well.
- Faster than random forest.

Weakness

- Very sensitive to new training data
- Overfitting can occur if parameters such as max depth is not controlled.
- Can be biased to features that is dominant (since it is greedy), reducing generalization.

- Worst case scenario needs to transverse until the maximum depth of tree when doing prediction.

Random Forest:

Strength:

- Random forest can pick random features, and thus, less likely chance to experience overfitting compared to decision tree.
- Since random forest can select random dataset and use it to generate random trees, random forest is less sensitive to unseen data (more generalized) compared to decision tree.

Weakness

- Can take a long time to compute, especially with large number of trees and tree depth.

In short, I will pick linear/logistic regression if the features and weight have linear relationship. If not, then I will take a further inspection and see my priority whether I prefer accuracy or time complexity. If the test requires high accuracy, then I will go for random forest, and if the model need to be efficient, I will go for decision tree.