

**School of InfoComm Technology**



**Machine Learning**

Diploma in Financial Informatics (FI)

Diploma in Information Technology (IT)

YR 2/3 (2021/22), Semester 4/6

**INDIVIDUAL ASSIGNMENT 2**

(40% of Machine Learning Module)

# Deadline for Submission:

**Presentation: 4th Feb 2022 (Friday), 2359 Hours**

**Report: 11th Feb 2022 (Friday), 2359 Hours**

| Student Name | : | Lim Jun Keat |
| --- | --- | --- |
| Student Number | : | S10205540F |

**Penalty for late submission:**

10% of the marks will be deducted every day after the deadline.

**NO** submission will be accepted after 18th Feb 2022, 23:59.

**Cover Page**[**:**](#_7gclge5924ha) **1**

[**Introduction**](#_vas9cgnc9fde) **3**

[**Classification Problem (HR Analytics)**](#_f6s4lhz7id8m) **4**

[Building the models](#_wac4r2kcnmnt) 5

[Logistic Regression](#_zdip1w33q0s) 5

[Decision Tree Classifier](#_jdzpalo0oelg) 7

[MLP Classifier](#_kqwtmm35o83a) 8

[XGBoost Classifier](#_5qjosonwc58r) 10

[Results of the initial Classification Models](#_5wz63l232gw4) 11

[Improving the models](#_1ysspfsujlkp) 12

[Logistic Regression](#_x12998w5agdn) 12

[Decision Tree Classifier](#_o7goa9ah4fkx) 13

[MLP Classifier](#_744vw6tsivlt) 15

[XGBoost Classifier](#_4pet87c61f4d) 16

[Results of the final Classification Models](#_il6e8ob7k1bv) 17

[**Regression Problem (Airbnb Analysis)**](#_ifyo049ay7fa) **18**

[Building the models](#_62afn3ccyvt9) 19

[Linear Regression](#_wb8ydxa7l4ql) 20

[Decision Tree Regressor](#_96ygx9bwqik0) 22

[MLP Regressor](#_bceb23m1ba1r) 23

[XGBoost Regressor](#_3d7cwcik32m9) 25

[Results of the initial Regression Models](#_rmo421gzhzol) 27

[Improving the models](#_l5zt6bvqo26x) 28

[Linear Regression](#_dy1g4aymston) 29

[Decision Tree Regressor](#_w58wxl4ro1u0) 30

[MLP Regressor](#_q0hs3q35ctj2) 31

[XGBoost Regressor](#_c4o9uw9fr5v9) 32

[Results of the final Regression Models](#_3nyk4zm3gaip) 33

[**Conclusion and Final Thoughts**](#_ms0bqo8dssg0) **34**

[**Appendix A**](#_m433u6qfx6ym) **35**

[**Appendix B**](#_olafm2fu54o6) **36**

# Introduction

The aim of this assignment is to utilize Machine Learning Models to solve both classification and regression problems in Python. 2 datasets will be used in this assignment, **hr\_data\_new.csv** and **listings\_new.csv**. Both sets of data have been cleansed and transformed back in Assignment 1.

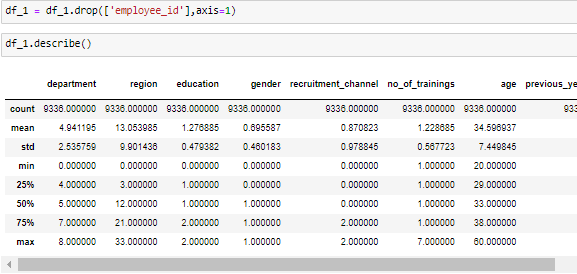
The dataset **hr\_data\_new.csv** contains employee personal information, educational background, past performance, promotion status and more. The information in the data set will be used in classification modelling, specifically to identify and predict employee’s that are more likely to be promoted using the other variables available in the dataset. The full data description is registered in Appendix A.

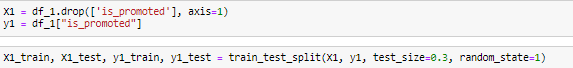
The dataset **listings\_new.csv** describes the listing activity and metrics from the year 2013 to 2019. The file contains data like the host’s information, condition of listed properties, listing reviews, listing location and more. This information will be utilised in regression modelling to make accurate predictions on the rental price of the listings. The full data description is registered in Appendix B.

This assignment will involve mainly 3 steps. The first step is to build the model using the default parameters. The second step is to evaluate and improve the performance of all models. The last step is to determine which model is the best and why, as well as provide further points of improvement.

# Classification Problem (HR Analytics)

For the classification problem, 4 models will be built, **Logistic Regression**, **Decision Tree Classifier**, **MLP Classifier** and **XGBoost Classifier**. The models will be built to predict the target feature, **is\_promoted**, using the other features. The models will first be built using their default parameters in order for us to evaluate the base performance of each model before making the necessary improvements and determining which model performs the best.





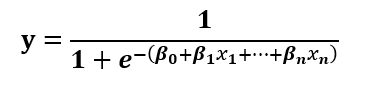
The dataset will first be read into a data frame called **df\_1**. The target feature will then be separated from the remaining features and be stored in a variable called **Y1** and **X1** respectively. **X1** and **Y1** will then be split using simple split, into 70% training data and 30% testing data.

## Building the models

As mentioned, the models will first be built using the default parameters to get the base performance. The models will then be evaluated using 2 methods. The first involves fitting the model with the training data and getting the accuracy of the model that has been fitted with the training data and the testing data. The next step is to use K-Fold Cross Validation to evaluate the model.

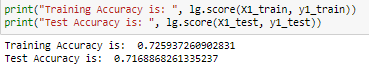
K-Fold Cross Validation involves splitting the dataset randomly into a number of equal-sized subsets. For this assignment, when performing K-Fold Cross Validation, the dataset will be split into 5 subsets. Each of the 5 subsets will be used as the testing data once while the rest will be used as the training data. This process will repeat until all 5 subsets have been used as the training data once. The accuracy K-Fold Cross Validation returns are more representative of the entire dataset than using simple split. This is because using simple split makes the model rely on the training data, thus adding another variability to the prediction performance. K-Fold Cross Validation ensures that the model is not overfitted and is able to predict new data and not only existing data that is used to train the model.

### Logistic Regression

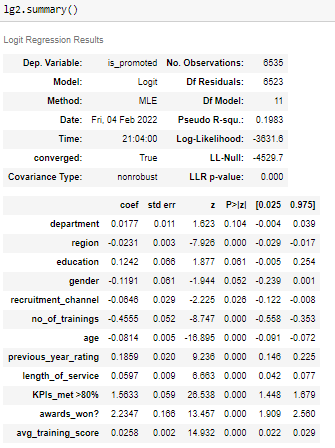


Logistic Regression is used to predict categorical target variables that have only 2 possible outcomes, either A or B or in our case, whether an employee is promoted or not. Logistic Regression works by using the logistic function (shown above) to convert the input values into another value between 0 and 1. The output is essentially a percentage probability that determines what value the predicted target variable is.



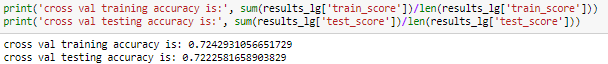


The training and testing accuracy of the Logistic Regression Model trained using simple split data

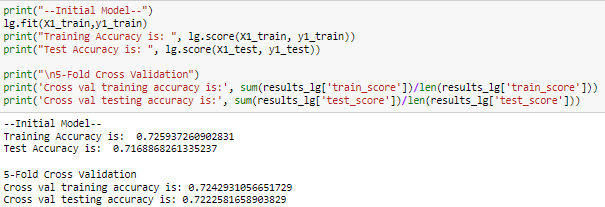


Stats model of the Logistic Regression Model

The above image shows the stats model of the Logistic Regression Model. The stats model displays the details of each feature and the relationship each feature has with the target variable. From the model, the attribute of the feature to look out for is **P>|z|**, or the P-score. The P-score essentially indicated whether a feature is a good indicator in predicting the target variable. The lower the P-score, the higher the confidence of the feature while higher P-scores mean the opposite. From the stats model, all of the features have a P-score of close to 0, which means that all the features are good in predicting whether an employee is promoted. Thus, none of the features have to be removed.

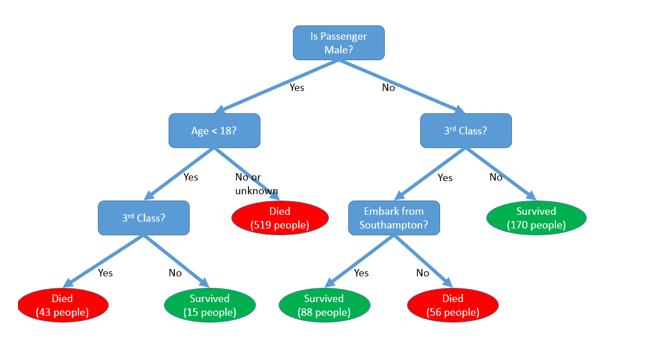


The training and testing accuracy of the Logistic Regression Model trained using K-Fold Cross Validation

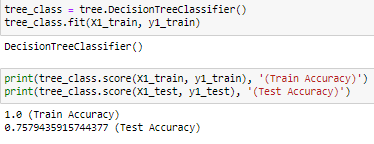


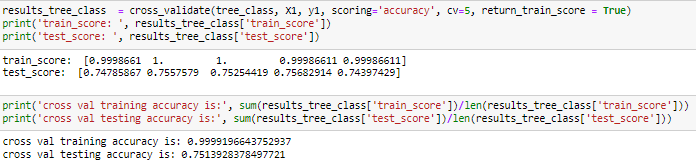
Results for the initial Logistic Regression Model

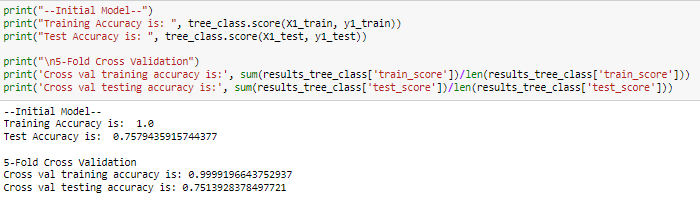
### Decision Tree Classifier



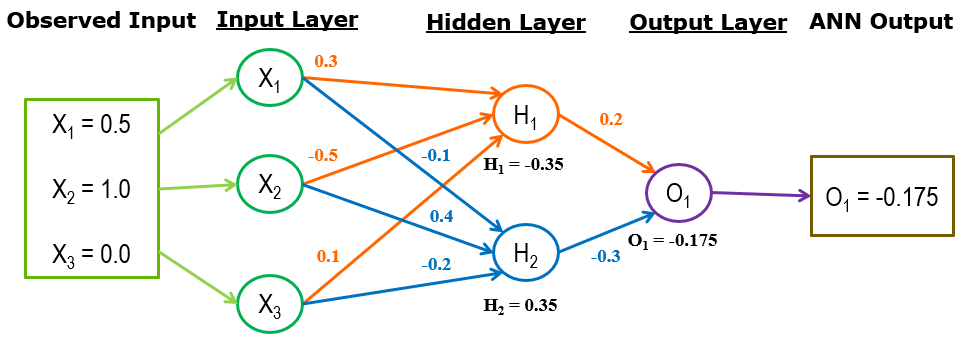
Decision Tree is a supervised learning algorithm that can be used for both regression and classification problems. Decision Trees work by building a model that predicts following the rules derived from the training data. Each rule is stored in a node which can be split into sub-nodes that contains more rules. The resulting structure looks like a tree hence the name of the algorithm.



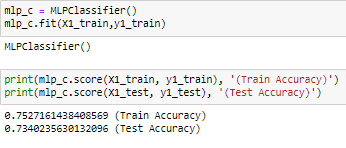


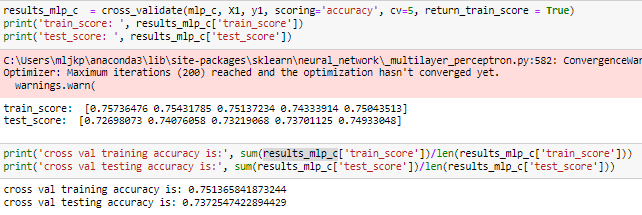


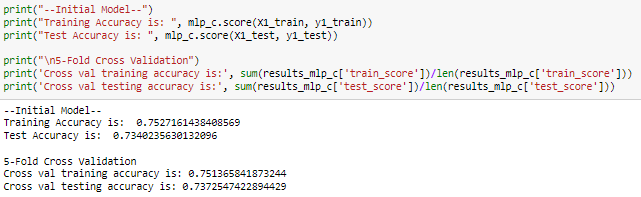
### MLP Classifier



MLP is one of the Artificial Neural Network (ANN) algorithms that can be used in both classification and regression problems. ANN is an information processing system that mimics the operations of the human brain, MLP introduces backwards propagation which allows the model to compute the errors of the predictions and update the network’s weight value. This process is repeated until the stopping criteria are met

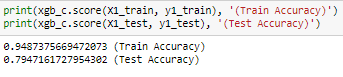


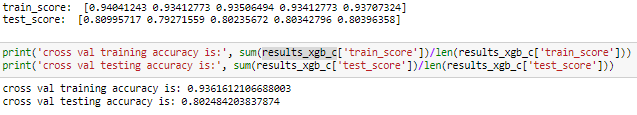


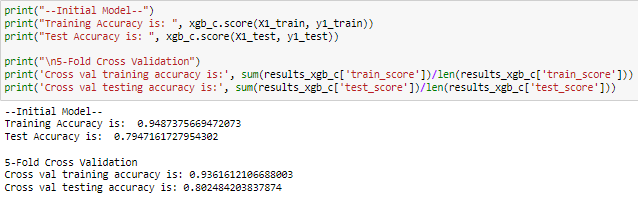


### XGBoost Classifier

XGBoost is a decision-tree based ensemble model which uses gradient boosting. Ensemble learning combines multiple models to produce one optimal prediction model. For classification problems, majority voting is adopted by ensemble learning to determine which is the final prediction based on what each model has voted or predicted. XGBoost uses gradient boosting, which aims to create new models that learn from the errors of the old models. This reduces the overall prediction errors over time.







## Results of the initial Classification Models

| Models | Training Accuracy | Test Accuracy | Cross Val Train Accuracy | Cross Val Test Accuracy |
| --- | --- | --- | --- | --- |
| Logistic Regression | 0.726 | 0.717 | 0.724 | 0.722 |
| Decision Tree Classifier | 1.0 | 0.758 | 1.0 | 0.751 |
| MLP Classifier | 0.753 | 0.734 | 0.751 | 0.737 |
| XGBoost Classifier | 0.948 | 0.794 | 0.936 | 0.802 |

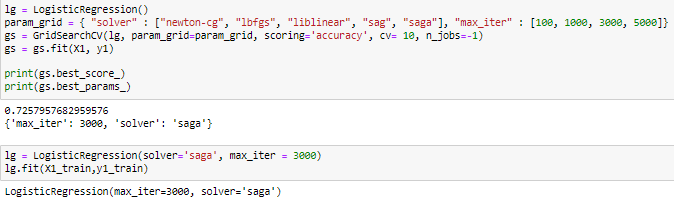
Row highlighted in green signifies the best result, red signifies worst result

Looking at the table of results above, XGBoost Classifier Model can be identified as the best performing model while Logistic Regression Model is the worst performing out of the 4. Although the XGBoost Classifier Model is the best performing, it is also overfitted like the Decision Tree Classifier Model. Overfitting is an error when the model fits exactly to the training data. This results in the model being unable to accurately predict outside data as the model is now only trained to exclusively predict the training data it has been fitted with. Thus, the training accuracy is higher than the testing accuracy for both models. This issue can be fixed in the next step which is the improve the performance of both the Decision Tree Classifier and Model XGBoost Classifier Model

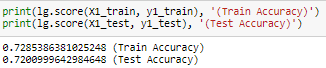
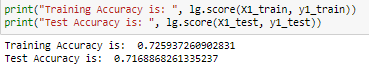
## Improving the models

In this step, the aim is to improve the performance of the model using GridSearch Cross-Validation. GridSearchCV is a tuning technique that tries every combination of hyperparameters passed into the model and evaluates the model using Cross-Validation. GridSearchCV then returns the best results or most accurate predictions. Using GridSearchCV is extremely convenient as it does not require us to manually test each combination of hyperparameters one by one. After identifying the combination of hyperparameters that gives the best performance, the improved model will once again be evaluated using simple split and K-Fold Cross-Validation.

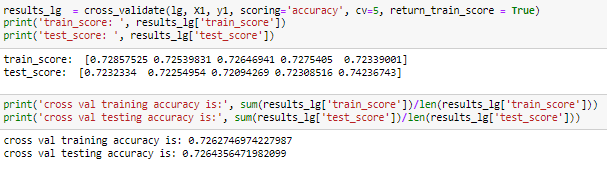
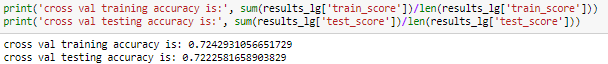
### Logistic Regression



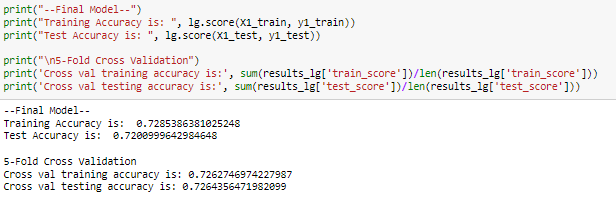
For Logistic Regression, 2 hyperparameters will be explored, **solver** and **max\_iter**. **solver** refers to the algorithm used in the optimization problem while **max\_iter** refers to the maximum number of iterations taken for the solvers to converge. After running GridSearchCV, the combination that returns the best results is the **saga solver** and a **max\_iter** of **3000**.



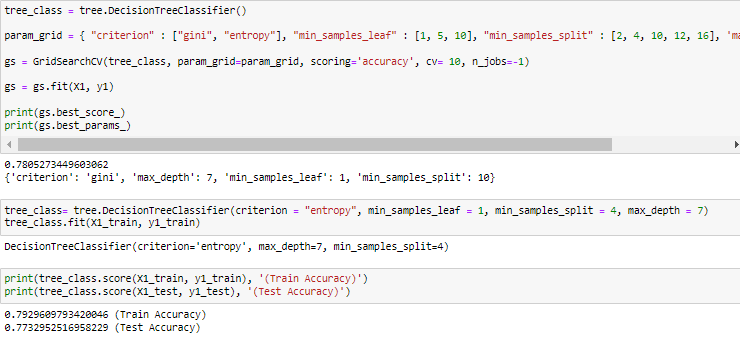
The improved model (right) has a slightly higher training and testing score than the old model (left)



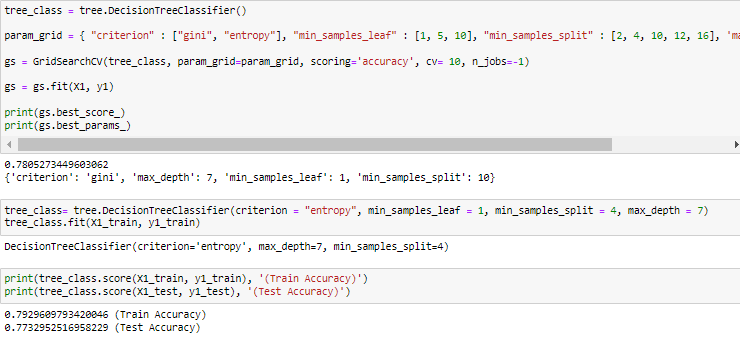
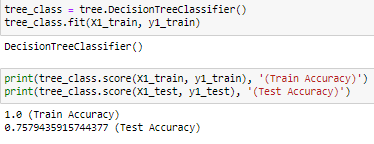
The improved model (right) also has a slightly higher cross-validation training and testing score than the old model (left)



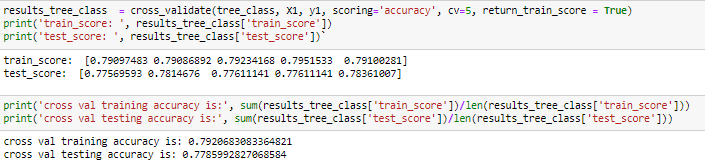
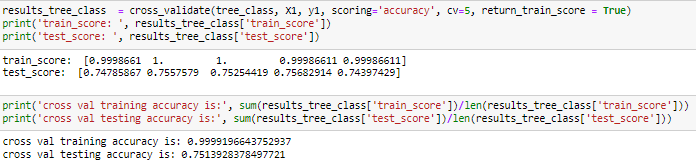
### Decision Tree Classifier



For Decision Tree Classifier, 4 hyperparameters will be explored, **criterion**, **min\_samples\_leaf**, **min\_samples\_split** and **max\_depth**. **criterion** is the function to measure the quality of a split. **min\_samples\_leaf** refers to the minimum number of samples required to be a leaf node, **min\_samples\_split** refers to the minimum number of samples required to split an internal node and **max\_depth** refers to the maximum depth of the tree. After running GridSearchCV, the combination that returns the best results is the **gini criterion**, **max\_depth** of **7**, **min\_samples\_leaf** of **1** and **min\_samples\_split** of **10**.

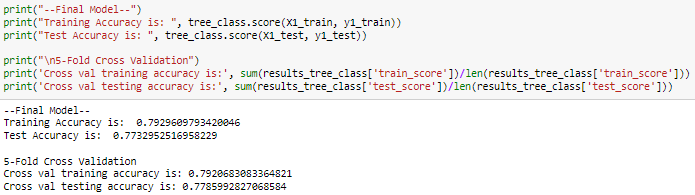


The improved model (right) has a lower training score but higher testing score than the old model (left)



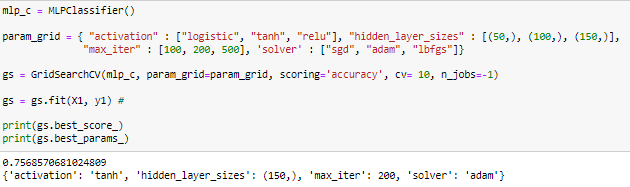
Similarly, the improved model (right) has a lower cross-validation training score and higher cross-validation testing score than the old model (left)

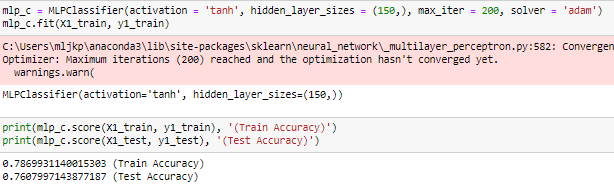
Comparing the result of the initial model with the improved model, it can be seen that the performance of the model has been significantly improved. The overall training score of the model has decreased and the overall testing score has increased. This shows that the model is no longer overfitted and is now able to predict new data more accurately.



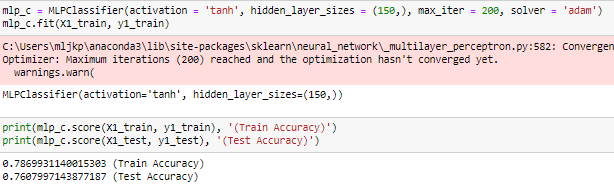
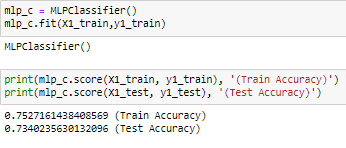
Results of the final Decision Tree Classifier Model

### MLP Classifier

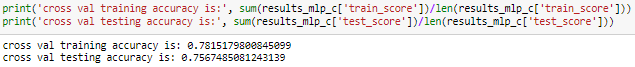
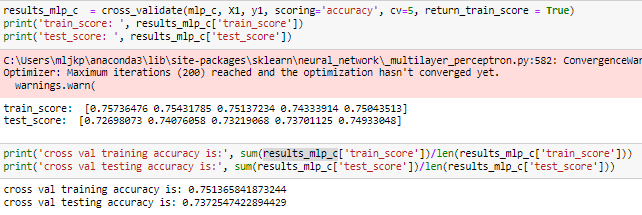




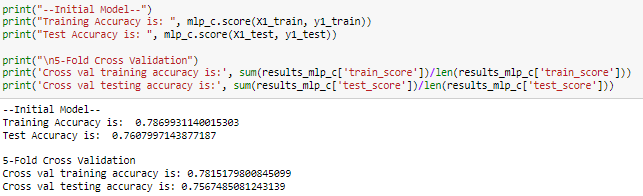
For MLP Classifier, 4 hyperparameters will be explored, **activation**, **hidden\_layer\_sizes**, **max\_iter** and **solver**. **activation** is the activation function for the hidden layer. **hidden\_layer\_sizes** refers to the ith element that represents the number of neurons in the ith hidden layer, **max\_iter** refers to the maximum number of iterations taken for the solvers to converge and **solver** is the solver for weight optimization. After running GridSearchCV, the combination that returns the best results is the **tanh activation**, **hidden\_layer\_sizes** of **150**, **max\_iter** of **200** and the **adam solver**.



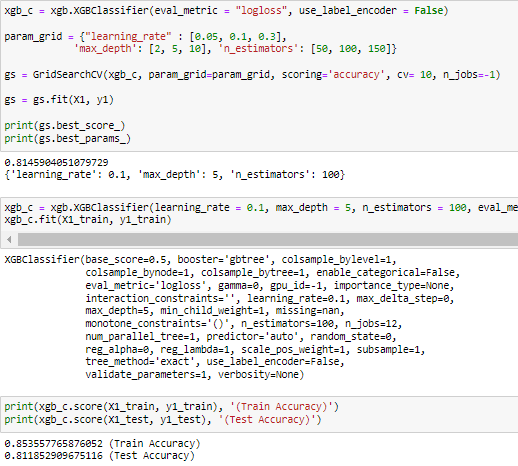
The improved model (right) has a higher training and testing score than the old model (left)



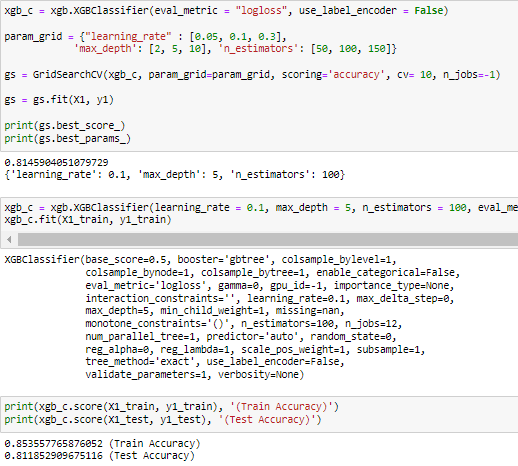
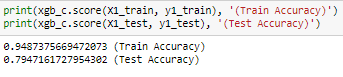
The improved model (right) also has a higher cross-validation training and testing score than the old model (left)



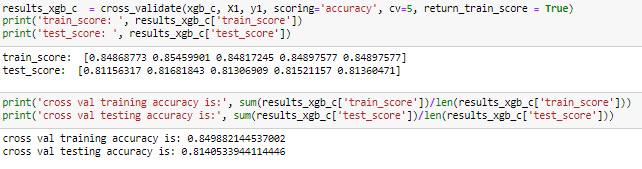
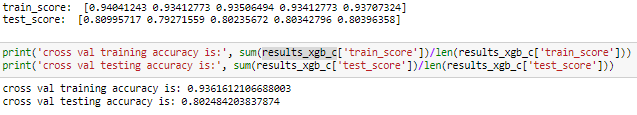
### XGBoost Classifier



For XGBoost Classifier, 3 hyperparameters will be explored, **learning\_rate**, **max\_depth**, and **n\_estimator**. **learning\_rate** is the size used to shrink the feature weights after each boosting process to prevent overfitting, **max\_depth** refers to the maximum depth of the tree and **n\_estimators** refers to the number of trees in the model. After running GridSearchCV, the combination that returns the best results is a model with a **learning\_rate** of **0.1**, **max\_depth** of **5** and **n\_estimators** of **100**.



The improved model (right) has a lower training score but slightly higher testing score than the old model (left)



Similarly, the improved model (right) has a lower cross-validation training score and higher cross-validation testing score than the old model (left)

## Results of the final Classification Models

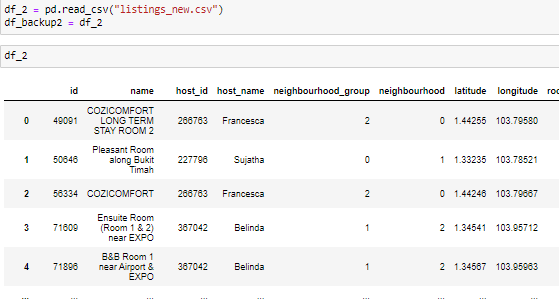
| Models | Training Accuracy | Test Accuracy | Cross Val Train Accuracy | Cross Val Test Accuracy |
| --- | --- | --- | --- | --- |
| Logistic Regression | 0.729 | 0.720 | 0.726 | 0.726 |
| Decision Tree Classifier | 0.793 | 0.773 | 0.792 | 0.779 |
| MLP Classifier | 0.787 | 0.761 | 0.782 | 0.757 |
| XGBoost Classifier | 0.850 | 0.812 | 0.850 | 0.814 |

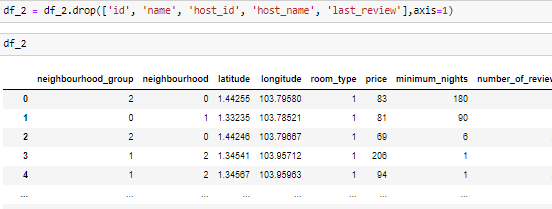
Row highlighted in green signifies the best result, red signifies worst result

Looking and comparing the results of the final models with the initial models, it can be seen that all of the models have been improved as the testing accuracy of all models have been increased. Furthermore, the Decision Tree Classifier and XGBoost Classifier Model are no longer overfitted and can predict outside data more accurately now. Overall, the best performing classifier model is still the XGBoost Classifier Model.

# Regression Problem (Airbnb Analysis)

For the Regression problem, the same 4 models will also be built, **Linear Regression**, **Decision Tree Regressor**, **MLP Regressor** and **XGBoost Regressor**. The models will be built to predict the target feature, **price**, using the other features. The models will first be built using their default parameters for us to evaluate the base performance of each model before making the necessary improvements and determining which model performs the best.









The dataset will first be read into a data frame called **df\_2**. The following features will also be removed from the data frame, **id**, **name**, **host\_id**, **host\_name** and **last\_review**. The reason those features are removed is that they are identifier variables and play no part in determining how a listing is priced. As for the **last\_review** feature, it is removed as the data in the **last\_review** feature is stored in the DateTime format and thus cannot be used to train the model. In assignment 1, the **last\_review** feature has already been used to create 2 new features **days\_open** and **months\_open** which can be used to train the model. The target feature will then be separated from the remaining features and be stored in a variable called **Y2** and **X2** respectively. **X2** and **Y2** will then be split using simple split, into 70% training data and 30% testing data.

## Building the models

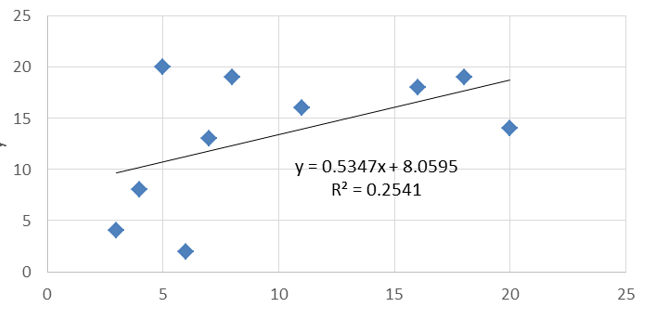
Similar to the classification problem, the regression models will also be built using the default parameters to get the base performance. The models will then be evaluated once again using 2 methods, simple split and K-Fold Cross-Validation. For the regression models, the models will be evaluated based on their Root Mean Squared Error (RMSE) and R^2 score.

RMSE is a measure of how spread out the predictions are from the actual data. So the lower the RMSE score of a model, the more accurate the model is. R2 is the opposite of RMSE as it shows how close the actual data is to the predicted data. So the higher the R2 score of a model, the more accurate the model is.

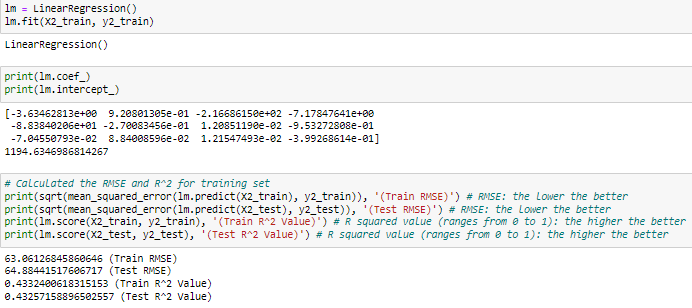
The reason that both RMSE and R2 is used to evaluate the model is that R2 is displayed on a 0 to 1 scale which is the same as the accuracy score of the classification models, thus making it easier to switch between the 2 problems.

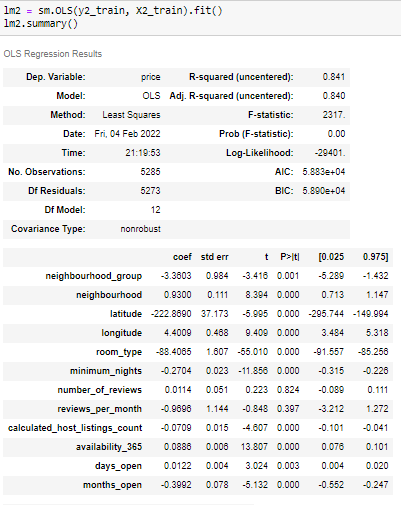
### Linear Regression

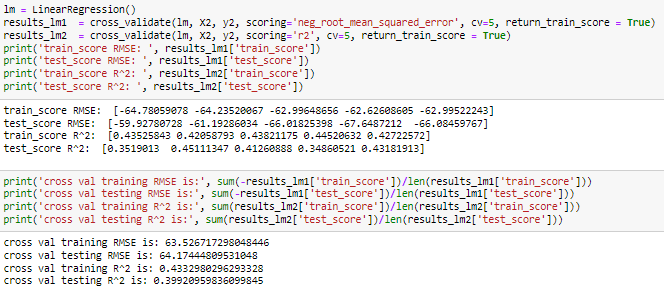


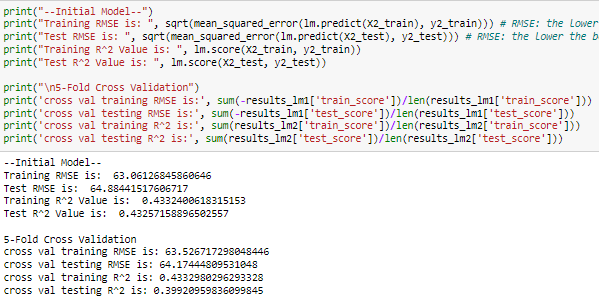


Linear Regression is used to predict continuous target variables that have infinite possible outcomes like in this case, the price of an Airbnb listing. Linear Regression works by identifying the line of best fit of the output with respect to the input.

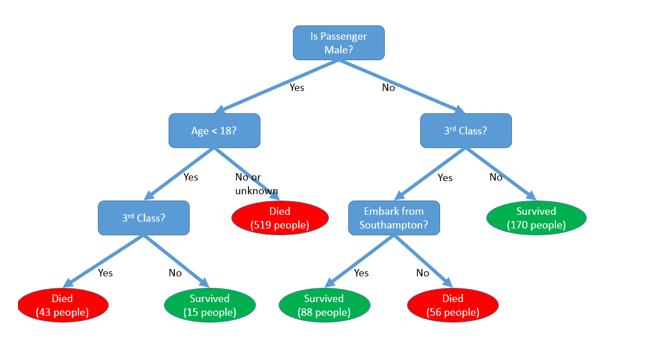




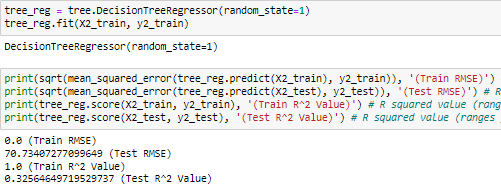


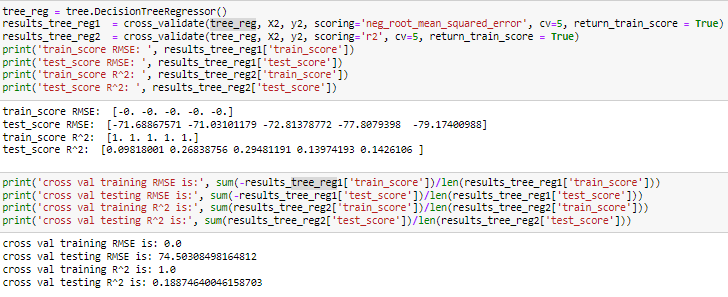


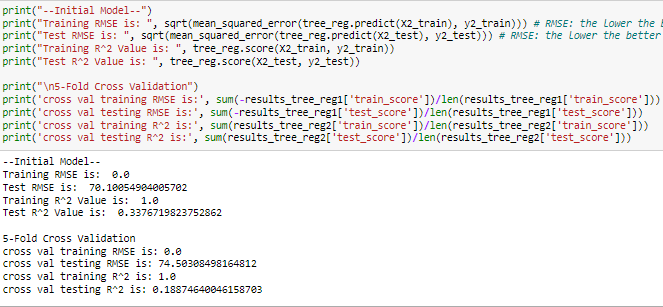
### Decision Tree Regressor



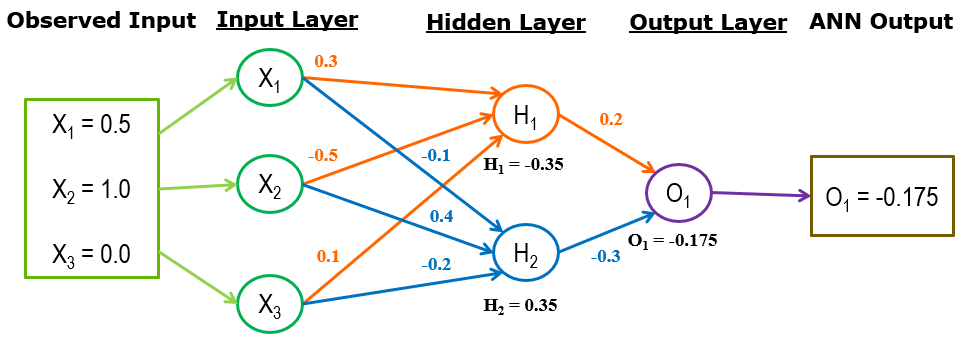
Decision Tree is a supervised learning algorithm that can be used for both regression and classification problems. Decision Trees work by building a model that predicts following the rules derived from the training data. Each rule is stored in a node which can be split into sub-nodes that contains more rules. The resulting structure looks like a tree hence the name of the algorithm.



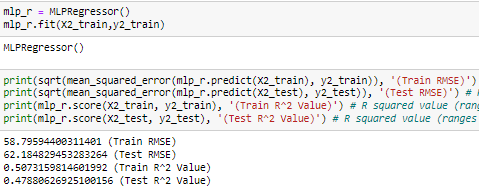


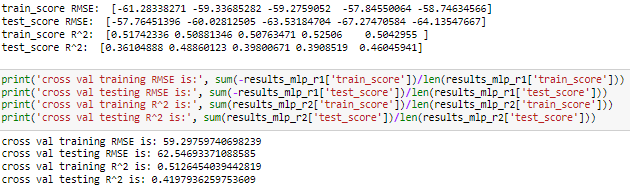
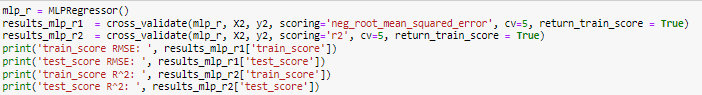


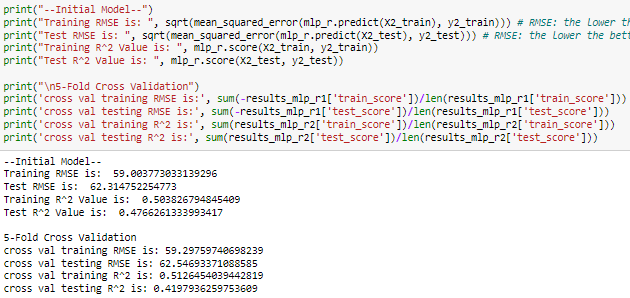
### MLP Regressor



MLP is one of the Artificial Neural Network (ANN) algorithms that can be used in both classification and regression problems. ANN is an information processing system that mimics the operations of the human brain, MLP introduces backwards propagation which allows the model to compute the errors of the predictions and update the network’s weight value. This process is repeated until the stopping criteria is met

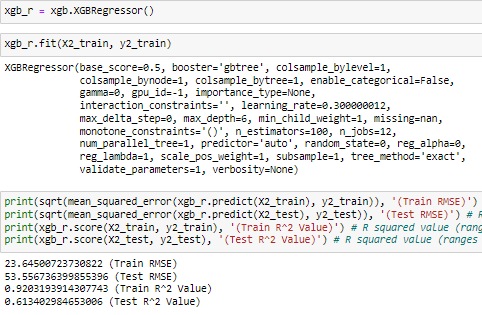


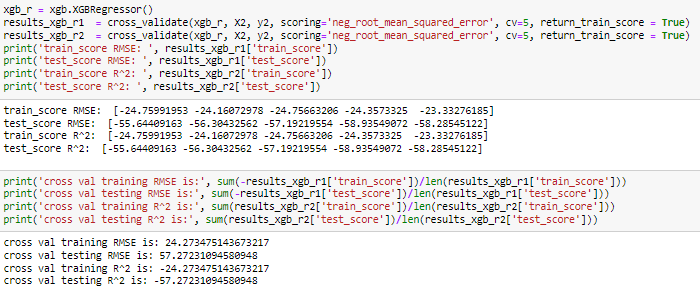


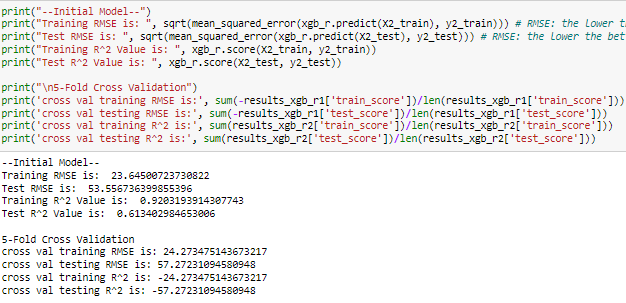


### XGBoost Regressor

XGBoost is a decision-tree based ensemble model which uses gradient boosting. Ensemble learning combines multiple models to produce one optimal prediction model. For regression problems, ensemble learning to determine the final prediction using the average predictions of all the regression models. XGBoost uses gradient boosting, which aims to create new models that learn from the errors of the old models. This reduces the overall prediction errors over time.







## Results of the initial Regression Models

RMSE

| Models | Training RMSE | Test RMSE | Cross Val Train RMSE | Cross Val Test RMSE |
| --- | --- | --- | --- | --- |
| Linear Regression | 63.061 | 64.884 | 63.527 | 64.174 |
| Decision Tree Regressor | 0.0 | 70.101 | 0.0 | 74.503 |
| MLP Regressor | 59.004 | 62.315 | 59.298 | 62.547 |
| XGBoost Regressor | 23.645 | 53.557 | 24.273 | 57.272 |

R2

| Models | Training R2 | Test R2 | Cross Val Train R2 | Cross Val Test R2 |
| --- | --- | --- | --- | --- |
| Linear Regression | 0.433 | 0.433 | 0.433 | 0.399 |
| Decision Tree Regressor | 1.0 | 0.338 | 1.0 | 0.189 |
| MLP Regressor | 0.504 | 0.477 | 0.513 | 0.420 |
| XGBoost Regressor | 0.920 | 0.613 | 0.917 | 0.520 |

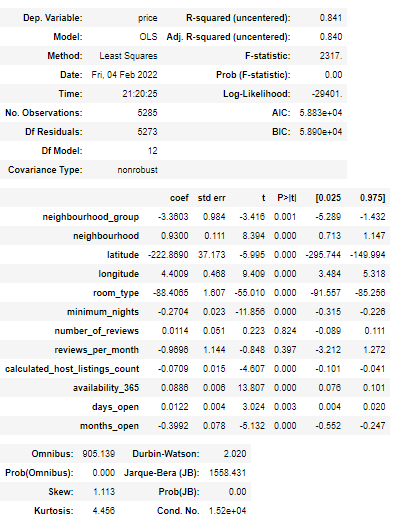
Row highlighted in green signifies best result, red signifies worst result

From the tables we can see that the results are all very low. The highest test R2 score is a mere 0.613 for simple split and 0.520 for K-Fold Cross-Validation. However, the pattern of the results is extremely similar to the results in the classification problem. The best performing model is once again XGBoost while the worst performing is Linear Regression. The Decision Tree Regressor Model and XGBoost model is also similarly overfitted. From the results, there are still many possible improvements that can be made to the models to improve their performance.

## Improving the models

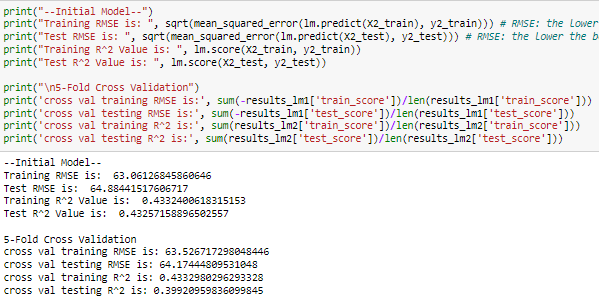
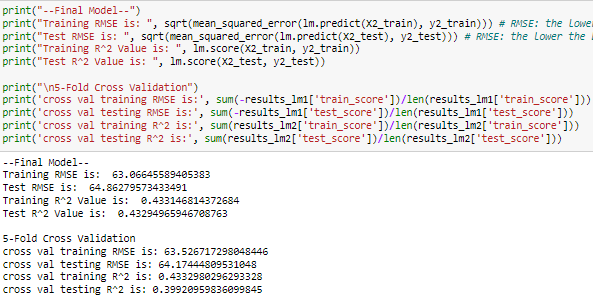
In this step, the aim is the same as the classification problem, to improve the performance of the model using GridSearch Cross-Validation. GridSearchCV is a tuning technique that tries every combination of hyperparameters passed into the model and evaluates the model using Cross-Validation. GridSearchCV then returns the best results or most accurate predictions. Using GridSearchCV is extremely convenient as it does not require us to manually test each combination of hyperparameters one by one. After identifying the combination of hyperparameters that gives the best performance, the improved regression models will once again be evaluated using simple split and K-Fold Cross-Validation.

### Linear Regression





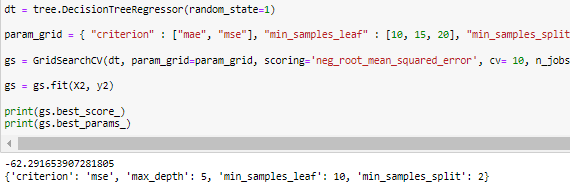
Since Linear Regression does not have many hyperparameters to modify, using GridSearchCV to try to improve the performance of the Linear Regression Model will not be very effective. Thus, instead of using GridSearchCV, the features with high P-score will be removed as they are not good indicators in predicting the target variable.

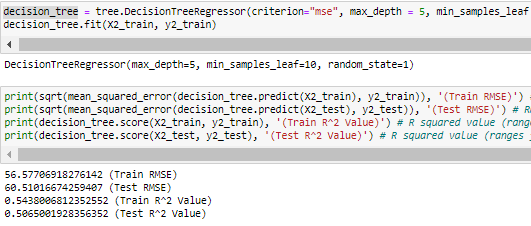
 

The improved model (right), old model (left)

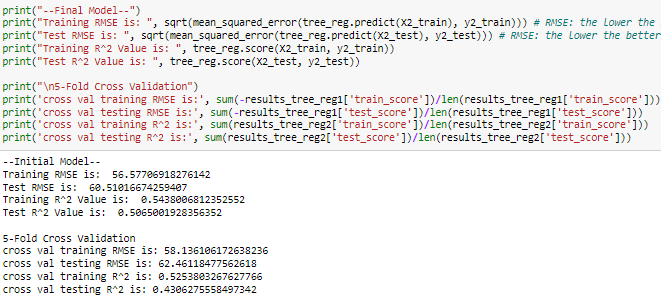
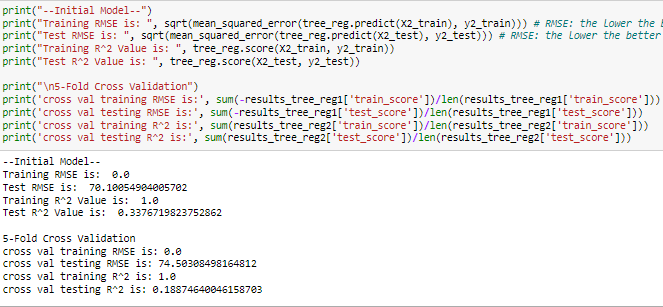
Comparing the results between the old model and the improved model, the accuracy only increased very slightly. This may be due to the lack of modifications made to the parameters of the model thus resulting in minimal changes to the performance of the model.

### Decision Tree Regressor





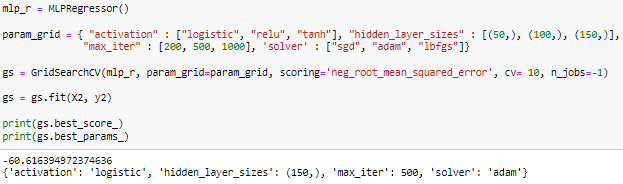
For Decision Tree Regressor, 4 hyperparameters will be explored, **criterion**, **min\_samples\_leaf**, **min\_samples\_split** and **max\_depth**. **criterion** is the function to measure the quality of a split. **min\_samples\_leaf** refers to the minimum number of samples required to be a leaf node, **min\_samples\_split** refers to the minimum number of samples required to split an internal node and **max\_depth** refers to the maximum depth of the tree. After running GridSearchCV, the combination that returns the best results is the **mse criterion**, **max\_depth** of **5**, **min\_samples\_leaf** of **10** and **min\_samples\_split** of **2**.

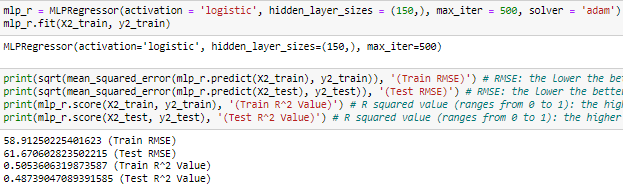


The improved model (right), old model (left)

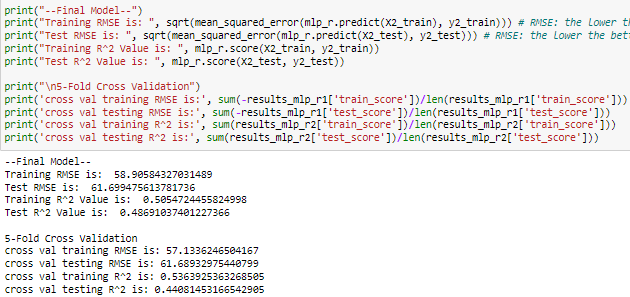
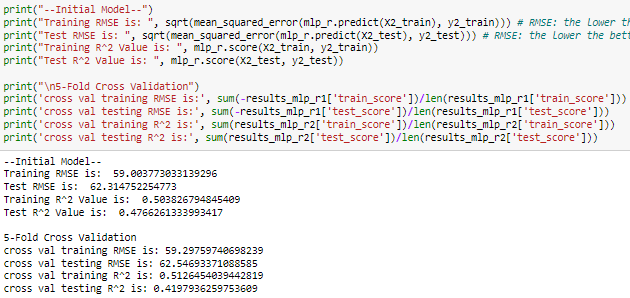
Comparing the results between the old model and the improved model, it can be seen that the model is no longer overfitted anymore. In the beginning, the testing R2 score of the initial model was extremely low compared to the perfect accuracy of the training R2 score. After using GridSearchCV to tune the model, the model now returns a much more favourable testing R2 score.

### MLP Regressor



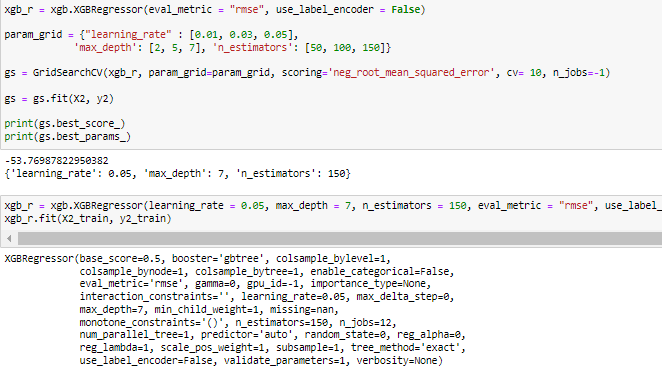


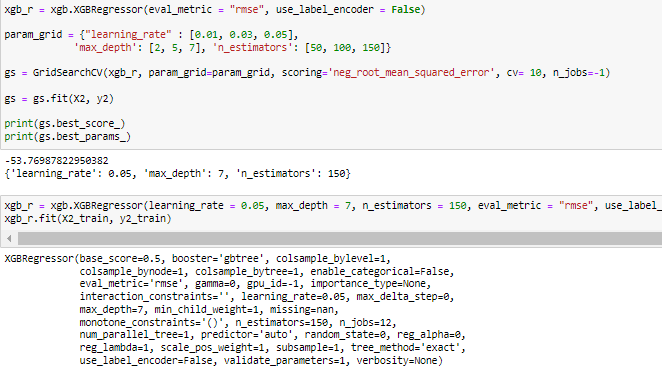
For MLP Regressor, 4 hyperparameters will be explored, **activation**, **hidden\_layer\_sizes**, **max\_iter** and **solver**. **activation** is the activation function for the hidden layer. **hidden\_layer\_sizes** refers to the ith element that represents the number of neurons in the ith hidden layer, **max\_iter** refers to the maximum number of iterations taken for the solvers to converge and **solver** is the solver for weight optimization. After running GridSearchCV, the combination that returns the best results is the **logistic activation**, **hidden\_layer\_sizes** of **150**, **max\_iter** of **500** and the **adam solver**.



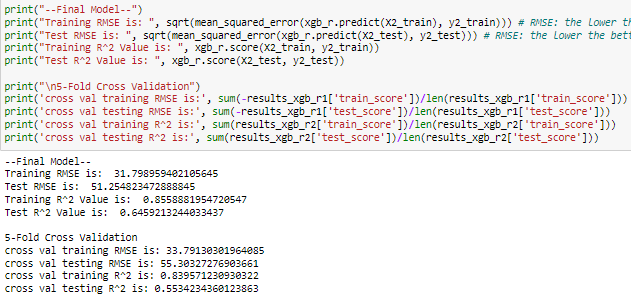
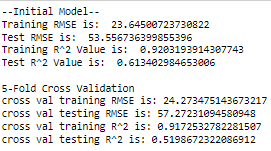
The improved model (right), old model (left). Similar to the Linear Regression Model, the improvement in performance is very slight.

### XGBoost Regressor





For XGBoost Regressor, 3 hyperparameters will be explored, **learning\_rate**, **max\_depth**, and **n\_estimator**. **learning\_rate** is the size used to shrink the feature weights after each boosting process to prevent overfitting, **max\_depth** refers to the maximum depth of the tree and **n\_estimators** refers to the number of trees in the model. After running GridSearchCV, the combination that returns the best results is a model with a learning\_rate of 0.05, **max\_depth** of **7** and **n\_estimators** of **150**.



The improved model (right), old model (left)

Looking at the difference between the performance of the initial model and the final model, the performance has improved slightly. Similar to the Decision Tree Regressor Model, this model was also initially overfitted. After making changes to the hyperparameters of the model, the testing R2 score of the model has increased slightly. However, the model is still overfitted. The overall training scores of the model is still significantly higher than the overall testing scores of the model. Thus, more improvements can still be made to the model.

## Results of the final Regression Models

RMSE

| Models | Training RMSE | Test RMSE | Cross Val Train RMSE | Cross Val Test RMSE |
| --- | --- | --- | --- | --- |
| Linear Regression | 63.066 | 64.862 | 63.527 | 64.174 |
| Decision Tree Regressor | 56.577 | 60.510 | 58.136 | 62.461 |
| MLP Regressor | 58.906 | 61.699 | 57.133 | 61.689 |
| XGBoost Regressor | 31.799 | 51.255 | 33.791 | 55.303 |

R2

| Models | Training R2 | Test R2 | Cross Val Train R2 | Cross Val Test R2 |
| --- | --- | --- | --- | --- |
| Linear Regression | 0.433 | 0.433 | 0.433 | 0.399 |
| Decision Tree Regressor | 0.544 | 0.507 | 0.525 | 0.431 |
| MLP Regressor | 0.505 | 0.487 | 0.536 | 0.441 |
| XGBoost Regressor | 0.856 | 0.646 | 0.840 | 0.553 |

Row highlighted in green signifies best result, red signifies worst result

Comparing the results between the initial and final models, almost all of the models have had their performance improved. However, the XGBoost Regressor Model can still be significantly improved as it is still overfitted. With that said, the XGBoost Regressor Model is still the best performing model out of the 4.

# Conclusion and Final Thoughts

After evaluating all of the models from both classification and regression problems, a clear pattern could be seen. The worst performing model in both problems has been the Logistic and Linear Regression Models while the best performing has always been the XGBoost Model. Although the Logistic and Linear Regression Model has consistently scored the lowest in terms of accuracy, the model that I believe has the worst performance is the MLP Models. Even though they did not have accuracy scores as low as the Logistic and Linear Models, they yielded extremely small improvements in results despite taking the longest to run. Running the GridSearchCV code for the MLP Models took close to an hour for each run and the improvements in performance were very small. Thus, using the MLP Models was the least effective and was not worth the time. The XGBoost Models performed the best in both problems. They returned the best results and did not take long for the code to run. This could be due to the ensemble learning and gradient boosting XGBoost adopts which aims to reduce the errors made in the new models made. Although the XGBoost Models were overfitted initially, with some tweaking and changes to the hyperparameters, the XGBoost Models has the potential to perform even better, returning more accurate predictions.

# Appendix A

Data Description of HR Data

| Variable | Values |
| --- | --- |
| employee\_id | Unique ID for employee |
| department | Analytics = 0, Finance = 1, HR = 2, Legal = 3, Operations = 4,  Procurement = 5, R&D = 6, Sales & Marketing = 7,  Technology = 8 |
| region | region\_1 = 0, region\_2 = 1, region\_3 = 2, region\_4 = 3,  region\_5 = 4, region\_6 = 5, 'region\_7 = 6, region\_8 = 7,  region\_9 = 8, region\_10 = 9, region\_11 = 10, region\_12 = 11, region\_13 = 12, region\_14 = 13, region\_15 = 14, region\_16 = 15, region\_17 = 16, region\_18 = 17, region\_19 = 18, region\_20 = 19, region\_21 = 20, region\_22 = 21, region\_23 = 22, region\_24 = 23, region\_25 = 24, region\_26 = 25, region\_27 = 26, region\_28 = 27, region\_29 = 28, region\_30 = 29, region\_31 = 30, region\_32 = 31, region\_33 = 32, region\_34 = 33 |
| education | Below Secondary = 0, Bachelor's = 1, Master's & above = 2 |
| gender | female = 0, male = 1 |
| recruitment\_channel | other = 0, referred = 1, sourcing = 2 |
| no\_of\_trainings | Values ranging from 1 to 7 |
| age | Values ranging from 0 to 1 (scaled values) |
| previous\_year\_rating | Values ranging from 0 to 5 |
| length\_of\_service | Values ranging from 0 to 1 (scaled values) |
| KPIs\_met >80% | Values ranging from 0 to 1 (Yes = 1, No = 0) |
| awards\_won? | Values ranging from 0 to 1 (Yes = 1, No = 0) |
| avg\_training\_score | Values ranging from 0 to 1 (scaled values) |
| is\_promoted | (Target) Recommended for promotion |

# Appendix B

Data Description of Airbnb Listings

| Variable | Values |
| --- | --- |
| id | Listing ID |
| name | Name of listing |
| host\_id | Host ID |
| host\_name | Name of the host |
| neighbourhood\_group | Central Region = 0, East Region = 1, North Region = 2, North-East Region = 3, West Region = 4 |
| neighbourhood | Woodlands = 0, Bukit Timah = 1, Tampines = 2,  Bedok = 3, Bukit Merah = 4, Newton = 5,  Geylang = 6, River Valley = 7, Jurong West = 8,  Rochor = 9, Queenstown = 10, Serangoon = 11,  Marine Parade = 12, Pasir Ris = 13, Toa Payoh = 14, Outram = 15, Punggol = 16, Tanglin = 17,  Hougang = 18, Kallang = 19, Novena = 20,  Downtown Core = 21, Bukit Panjang = 22,  Singapore River = 23, Orchard = 24, Ang Mo Kio = 25, Bukit Batok = 26, Museum = 27,  Sembawang = 28, Choa Chu Kang = 29,  Central Water Catchment = 30, Sengkang = 31,  Clementi = 32, Jurong East = 33, Bishan = 34,  Yishun = 35, Mandai = 36, Southern Islands = 37,  Sungei Kadut = 38, Western Water Catchment = 39, Tuas = 40, Marina South = 41, Lim Chu Kang = 43 |
| latitude | Latitude coordinates |
| longtitude | Longtitude coordinates |
| room\_type | Entire home/apt = 0, Private room = 1, Shared room = 2 |
| price | (Target) daily rental price in dollars |
| minimum\_nights | Values ranging from 0 to 1 (scaled values) |
| number\_of\_reviews | values ranging from 0 to 1 (scaled values) |
| last\_review | Date ranging from |
| reviews\_per\_month | Values ranging from 0 to 13 |
| calculated\_host\_listings\_count | values ranging from 0 to 1 (scaled values) |
| availability\_365 | values ranging from 0 to 1 (scaled values) |
| days\_open | values ranging from 0 to 1 (scaled values) |
| months\_open | values ranging from 0 to 1 (scaled values) |

**[END]**