UNIVERSITI TUNKU ABDUL RAHMAN
Wholly owned by UTAR Education Foundation
(Co. No. 578227-M)
DU012(A)

UEMH3073 Artificial Intelligence

Practical 2 Report

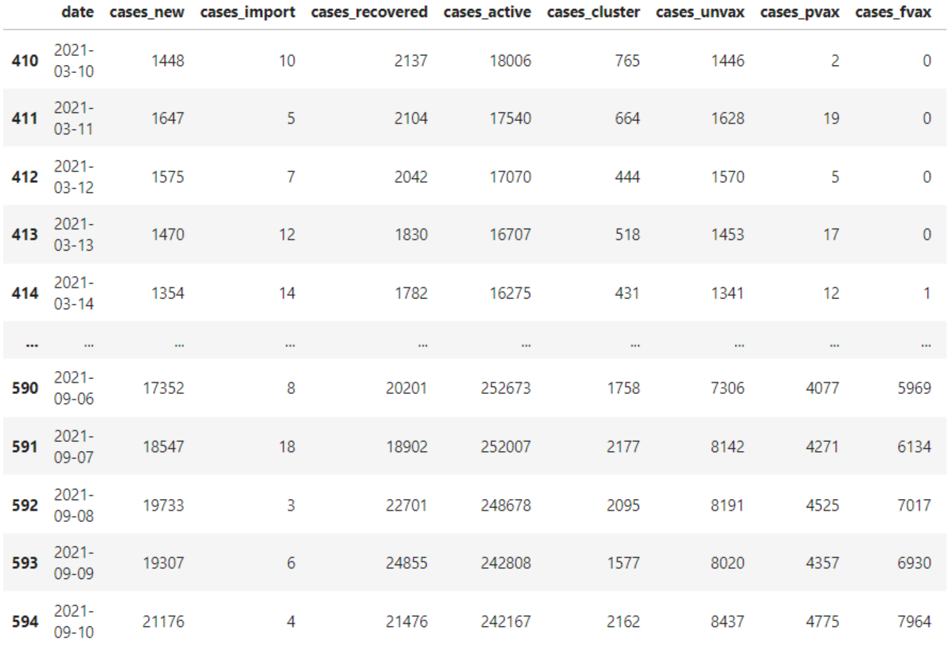
Supervised Learning

2024

Student ID	Student Name	Practical Group	Program
2202489	Ng Yen Keat	P1	MH
2200574	Ong Hong Qing	P1	MH
2102457	Kam York Yong	P1	MH
2202291	Goh Guang Yi	P1	MH

Table of Contents

1 Feature, Target selection and preprocessing method



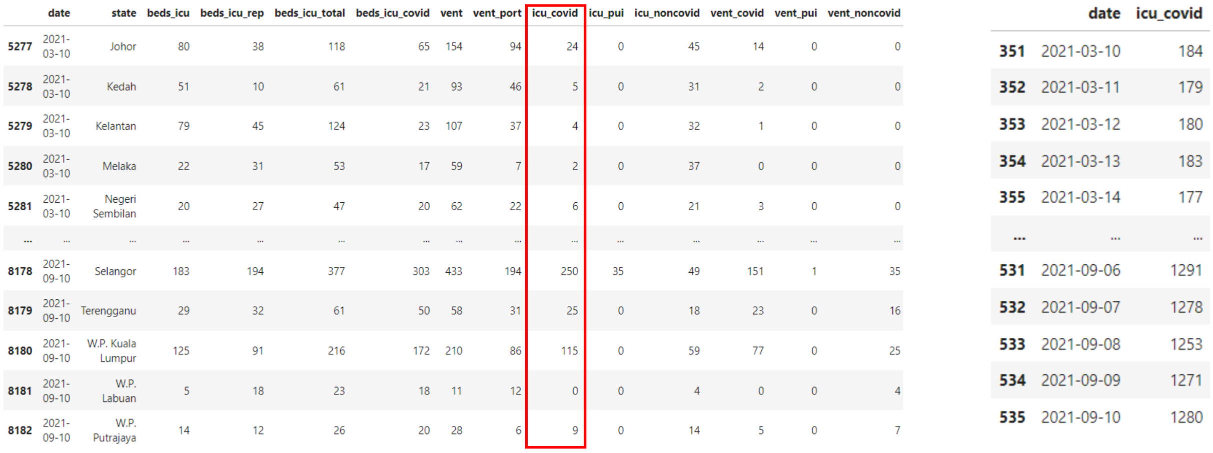
	date	cases_new	cases_import	cases_recovered	cases_active	cases_cluster	cases_unvax	cases_pvax	cases_fvax
410	2021-03-10	1448	10	2137	18006	765	1446	2	0
411	2021-03-11	1647	5	2104	17540	664	1628	19	0
412	2021-03-12	1575	7	2042	17070	444	1570	5	0
413	2021-03-13	1470	12	1830	16707	518	1453	17	0
414	2021-03-14	1354	14	1782	16275	431	1341	12	1
...	...	...	...	...	...	...	...	...	...
590	2021-09-06	17352	8	20201	252673	1758	7306	4077	5969
591	2021-09-07	18547	18	18902	252007	2177	8142	4271	6134
592	2021-09-08	19733	3	22701	248678	2095	8191	4525	7017
593	2021-09-09	19307	6	24855	242808	1577	8020	4357	6930
594	2021-09-10	21176	4	21476	242167	2162	8437	4775	7964

Figure 1.1: Some of the input features shown from 10/3/2021 to 10/9/2021 in "case_malaysia.csv".

Discussion:

For the data range, we have chosen the datasets covering the period from 10/3/2021 up to 10/9/2021 (six consecutive months). The period was selected because it marks the onset of covid-19 diseases outbreak in Malaysia, which is a critical time frame for our analysis. By focusing on the period, we aim to make meaningful insight and thorough investigation into the datasets collected.

For our model training, we decided to include all the columns from the "case_malaysia.csv" as input features. The rationale behind the decision is to not be missing any potentially important features. While some features might not seem important at first glance, they could still contribute valuable information that may influence the model's performance and prediction.



	date	state	beds_icu	beds_icu_rep	beds_icu_total	beds_icu_covid	vent	vent_port	icu_covid	icu_pui	icu_noncovid	vent_covid	vent_pui	vent_noncovid
5277	2021-03-10	Johor	80	38	118	65	154	94	24	0	45	14	0	0
5278	2021-03-10	Kedah	51	10	61	21	93	46	5	0	31	2	0	0
5279	2021-03-10	Kelantan	79	45	124	23	107	37	4	0	32	1	0	0
5280	2021-03-10	Melaka	22	31	53	17	59	7	2	0	37	0	0	0
5281	2021-03-10	Negeri Sembilan	20	27	47	20	62	22	6	0	21	3	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8178	2021-09-10	Selangor	183	194	377	303	433	194	250	35	49	151	1	35
8179	2021-09-10	Terengganu	29	32	61	50	58	31	25	0	18	23	0	16
8180	2021-09-10	W.P. Kuala Lumpur	125	91	216	172	210	86	115	0	59	77	0	25
8181	2021-09-10	W.P. Labuan	5	18	23	18	11	12	0	0	4	0	0	4
8182	2021-09-10	W.P. Putrajaya	14	12	26	20	28	6	9	0	14	5	0	7

	date	icu_covid
351	2021-03-10	184
352	2021-03-11	179
353	2021-03-12	180
354	2021-03-13	183
355	2021-03-14	177
...	...	...
531	2021-09-06	1291
532	2021-09-07	1278
533	2021-09-08	1253
534	2021-09-09	1271
535	2021-09-10	1280

Figure 1.2: (Left) some of the output features shown in “icu.csv” from 10/3/2021 to 10/9/2021, (Right) sum of the icu_covid features group by to the same date.

Discussion:

For the target of the prediction in our model training, we choose the “icu_covid” in the file “icu.csv” from the time of 10/3/2021 to 10/9/2021, which is the same as the input features. The number of cases in different states are added together to show the total number of “icu_covid” cases in the respective date.

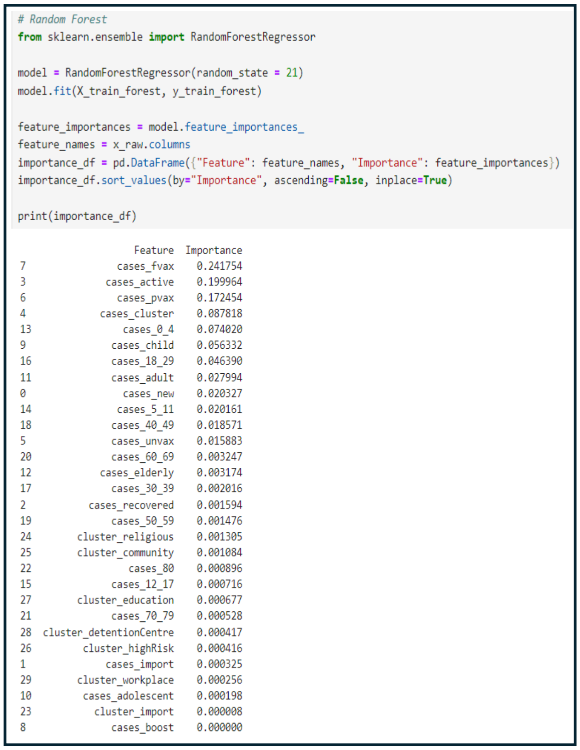
The reason for choosing the “icu_covid” as target is to clearly show the number of Covid ICU patients affected by the features in the data set. Besides, targeting to predict the Covid ICU patient can show the intensity level of covid situation during the time from 10/3/2021 to 10/9/2021 and determine the speed and pattern of Covid virus infection.

Min-max scaling

Before the data set is applied to the model, normalization of data is needed to apply to the data set. In this model, min-max scaling is used to normalize the data in the features to transform the data to a specific range between 0 and 1 (Singh, 2023). Scaling is important in machine learning algorithms as the result of the learning process will be better when the data is normalized. With this technique, we can prevent any feature from dominating the learning process. This process is important to the algorithm that are sensitive to the scale of data. Moreover, scaling of data increases the speed of convergence of the training process and has made the visualization of result clearer.

2 Types of Algorithms

2.1 Random Forest



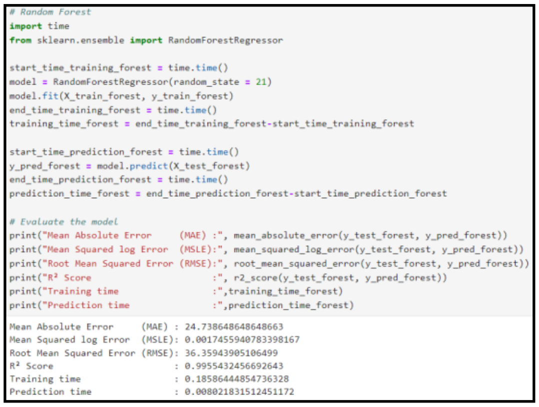
```
# Random Forest
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(random_state = 21)
model.fit(X_train_forest, y_train_forest)

feature_importances = model.feature_importances_
feature_names = x_raw.columns
importance_df = pd.DataFrame({"Feature": feature_names, "Importance": feature_importances})
importance_df.sort_values(by="Importance", ascending=False, inplace=True)

print(importance_df)
```

```
                   Feature  Importance
7               cases_fvax    0.241754
3             cases_active    0.199964
6               cases_pvax    0.172454
4            cases_cluster    0.087818
13               cases_0_4    0.074020
9              cases_child    0.056332
16             cases_18_29    0.046390
11             cases_adult    0.027994
0                cases_new    0.020327
14              cases_5_11    0.020161
18             cases_40_49    0.018571
5              cases_unvax    0.015883
20             cases_60_69    0.003247
12           cases_elderly    0.003174
17             cases_30_39    0.002016
2          cases_recovered    0.001594
19             cases_50_59    0.001476
24       cluster_religious    0.001305
25       cluster_community    0.001084
22                cases_80    0.000896
15             cases_12_17    0.000716
27       cluster_education    0.000677
21             cases_70_79    0.000528
28  cluster_detentionCentre    0.000417
26        cluster_highRisk    0.000416
1             cases_import    0.000325
29       cluster_workplace    0.000256
10        cases_adolescent    0.000198
23          cluster_import    0.000008
8              cases_boost    0.000000
```



```
# Random Forest
import time
from sklearn.ensemble import RandomForestRegressor

start_time_training_forest = time.time()
model = RandomForestRegressor(random_state = 21)
model.fit(X_train_forest, y_train_forest)
end_time_training_forest = time.time()
training_time_forest = end_time_training_forest-start_time_training_forest

start_time_prediction_forest = time.time()
y_pred_forest = model.predict(X_test_forest)
end_time_prediction_forest = time.time()
prediction_time_forest = end_time_prediction_forest-start_time_prediction_forest

# Evaluate the model
print("Mean Absolute Error     (MAE) :", mean_absolute_error(y_test_forest, y_pred_forest))
print("Mean Squared log Error  (MSLE):", mean_squared_log_error(y_test_forest, y_pred_forest))
print("Root Mean Squared Error (RMSE):", root_mean_squared_error(y_test_forest, y_pred_forest))
print("R² Score                      :", r2_score(y_test_forest, y_pred_forest))
print("Training time                 :",training_time_forest)
print("Prediction time               :",prediction_time_forest)
```

```
Mean Absolute Error     (MAE) : 24.738648648648663
Mean Squared log Error  (MSLE): 0.0017455940783398167
Root Mean Squared Error (RMSE): 36.35943905106499
R² Score                      : 0.9955432456692643
Training time                 : 0.18586444854736328
Prediction time               : 0.008021831512451172
```

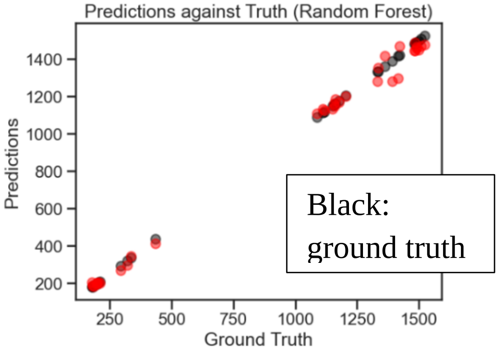


Figure 2.1: Showing the result of Random Forest method (random seed = 21) of correlations values of 30 input features and the prediction result with graph.

Training time (seconds, s)					
Trial1	Trial2	Trial3	Trial4	Trial5	Average
0.2088	0.1871	0.2150	0.2016	0.1859	0.1997

Table 2.1a: Training time for Random Forest method.

Prediction time (seconds, s)					
Trial1	Trial2	Trial3	Trial4	Trial5	Average
0.0080	0.0079	0.0075	0.0106	0.0080	0.0084

Table 2.1b: Prediction time for Random Forest method.

Discussion:

Random forest is an algorithm that combines the output of multiple decision trees with a single accurate and stable prediction result. Random forest algorithm adds additional randomness, easy to use, flexible and it also reduces the risk of overfitting, bias, and overall variance, resulting in more precise predictions (Built In., n.d.).

In this model, we used random seed with a value of 21 to ensure the same decision tree is selected every time we run the code. If the random seed is not selected, the output result will vary slightly in every trial. The prediction against truth graph is shown above. We can observe the R-squared value to determine the fitness of the model to the observed data and the performance of model to predict the data (Minitab.com, 2024). A higher value of R-squared value indicates a better model. From the result, the R-squared value is approximately 0.996 (correct to three decimal places), which is considered high value and hence random forest is a good model in predicting the target.

Besides, mean square error can be an indicator to determine the fitness of model in the prediction. From the result, the mean square logarithm error for random forest is approximately 0.0017 (correct to 4 decimal places) which is small enough to show that random forest is suitable for the prediction. Mean squared error calculates the difference between the true value and the predicted value from the model. Hence, a small value means a small difference between the predicted value and true value. The graph of prediction against truth shows that the predicted target is close to the true target.

For the training time and prediction time, we processed the model for 5 trials and got the average time. The training time for random forest algorithm is 0.1997s while the prediction time is 0.0084s.

2.2 Linear Regression

Training time (seconds, s)					
Trial1	Trial2	Trial3	Trial4	Trial5	Average
0.0010	0.0014	0.0010	0.0028	0.0011	0.0015

Table 2.2a: Training time for linear regression method.

Prediction time (seconds, s)					
Trial1	Trial2	Trial3	Trial4	Trial5	Average
0.0000	0.0005	0.0000	0.0010	0.0000	0.0003

Table 2.2b: Prediction time for linear regression method.

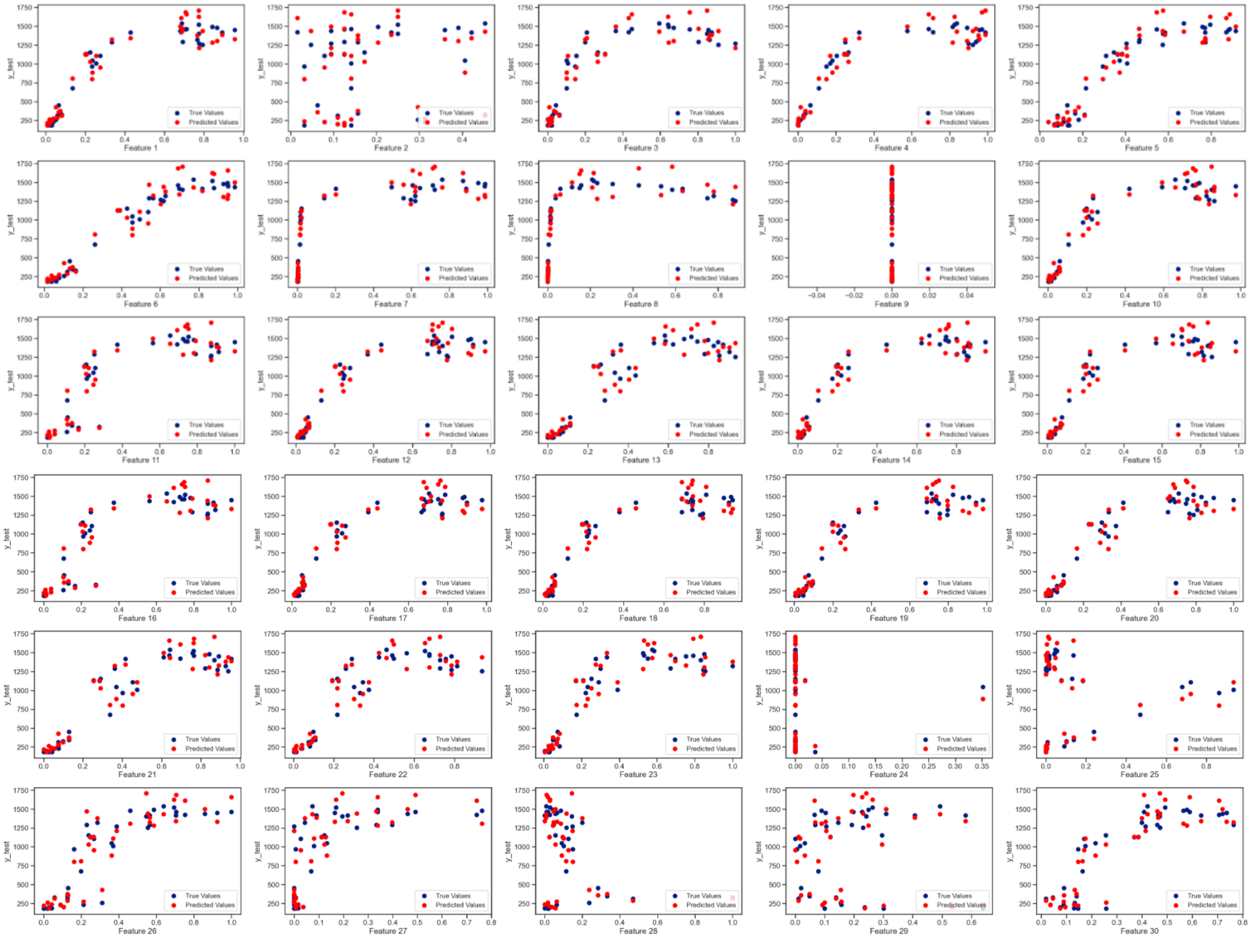
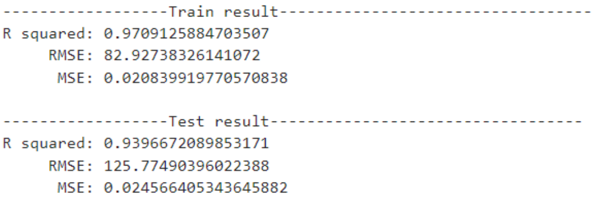


Figure 2.2: Scatter plot matrix diagram of 30 distinct input features of predicted values (red dots) against ground truth (blue dots) graph.



```
------------------Train result----------------------------------
R squared: 0.9709125884703507
     RMSE: 82.92738326141072
      MSE: 0.020839919770570838

------------------Test result----------------------------------
R squared: 0.9396672089853171
     RMSE: 125.77490396022388
      MSE: 0.024566405343645882
```

Figure 2.3: Train and Test result for R squared, RMSE and MSE analysis in Linear regression method (random seed = 22).

Discussion:

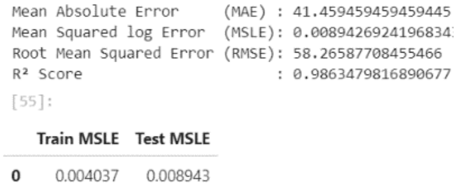
Linear regression is a simple yet powerful algorithm that predict the value of a dependent variable based on one or more independent variables. (Amazon Web Services, Inc., n.d.). Linear regression is relatively simple and easy to generate predictions by interpret mathematical formula. The variable that we wish to predict is called dependent variable while the variable that we used to predict the other variable is called independent variable. (IBM， 2024). In its simplest form, linear regression can be visualized on a 2D graph, where the horizontal axis represents the independent variable, and the vertical axis represents the dependent variable. A graph can be plotted using mathematical equation for example a simple equation, $y = mx + c$.

However, linear regression has great limitation of flexibility as it assumes the linear relationship between dependent variable and independent variable. Besides, it also prone to mistakes and outliers which means it will easily be affected by variable that has abnormal distance from other values (Indeed Career Guide., n.d.).

From the result, the mean square error is 0.025 (correct to 3 decimal places) and the R-squared value is 0.940 (correct to 3 decimal places). The MSE being small suggests that the predictions are close to the true values, while the high R-squared value indicates that a large proportion of the variance in the dependent variable is explained by the independent variables.

In Figure 2.2, as the input features are more than one, so the algorithm performs multiple linear regression, which consider all features simultaneously. The relationship between each feature and the target variable when viewed separately is illustrated by the individual scatter plot, but the prediction might differ if all the feature are combined. The training time for linear regression is 0.0015s while the test time is 0.0003s, which are considered relatively fast compared to the other methods.

2.3 KNN Regression

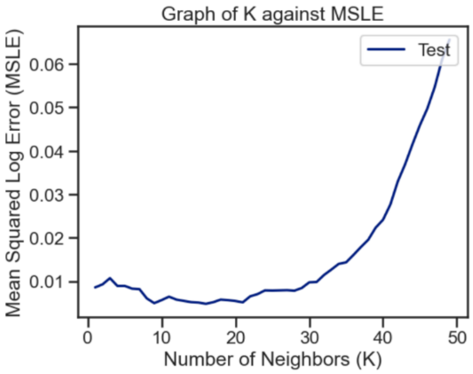


```
Mean Absolute Error     (MAE) : 41.459459459459445
Mean Squared log Error  (MSLE): 0.0089426924196834
Root Mean Squared Error (RMSE): 58.26587708455466
R² Score                      : 0.9863479816890677
[55]:
```

	Train MSLE	Test MSLE
0	0.004037	0.008943

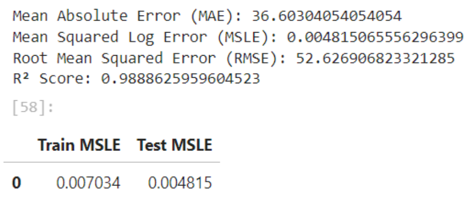
KNN regression is first initialized with 5 neighbors, where only 5 nearest points are to be considered by the model to make predictions. The prediction is made on the training set and test set for the ease of evaluation of the model's performance. The result shows that different evaluations are made to the model. The Mean Absolute Error (MAE) measures the average error whereas the Root Mean Squared Error (RMSE) measures the square root of the average squared difference. The value of R^2 score indicates how well the model fits the data. The value obtained shows that the predictions are highly accurate, and highly effective in predicting the target variable.

The Mean Squared Logarithmic Error (MSLE) values for both sets are also determined. These values measure the logarithmic difference between the actual and predicted values, with lower values indicate better predictions. The train MSLE shows that the model captures the training data effectively, only with a small difference. On the other hand, the test MSLE shows that the prediction is not as accurate as the training data. This is due to the test data being used to evaluate the generalization of the model.



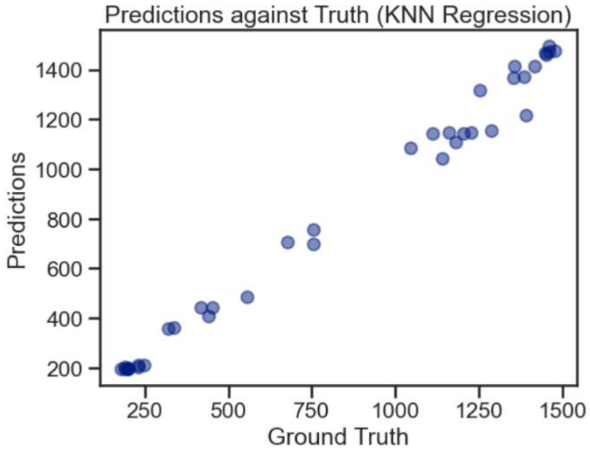
```
Best k: 16
```

A loop that iterates over K values ranging from 1 to 49 is then used to determine the best K value for the prediction. This value is determined by selecting the value that provides the lowest MSLE value after evaluating the model for each value and 16 is obtained as the best K value.



```
Mean Absolute Error (MAE): 36.60304054054054
Mean Squared Log Error (MSLE): 0.004815065556296399
Root Mean Squared Error (RMSE): 52.626906823321285
R² Score: 0.9888625959604523
[58]:
```

	Train MSLE	Test MSLE
0	0.007034	0.004815



Training time (seconds, s)					
Trial1	Trial2	Trial3	Trial4	Trial5	Average
0.0071	0.0035	0.0057	0.0035	0.0030	0.0046

Table 2.3a: Training time for KNN regression method for k = 5.

Prediction time (seconds, s)					
Trial1	Trial2	Trial3	Trial4	Trial5	Average
0.0075	0.0080	0.0082	0.0070	0.0063	0.0074

Table 2.3b: Prediction time for KNN regression method for k = 5.

The training time and prediction time both have operation times less than 0.01 seconds, and the sum of both times is 0.012 seconds, which is considered fast. For a small-sized dataset, such as the features used in this lab session, this operation time is fine and will not delay the process. A fast operation time also indicates that the model is scalable and is able to handle more complex scenarios or larger datasets according to the circumstances without causing significant delays.

2.4 Neural Network

Discussion:

Neural networks are very useful tools for tasks like regression and classification. Its performance can be highly affected by epochs, optimisers, hidden layers, learning rate, and batch size. Increasing the number of epochs will reduce loss because the model learns more from the data. However, using too many epochs will cause overfitting, which happens when the model performs well on training data but badly on new data. Optimisers are also very important to how neural network changes its weights to reduce the loss function. Different optimisers have major impacts on the speed and quality of learning. Optimisers like Adam, SGD, and RMSProp control how the model's weights are updated. One example, Adam adapts the learning rate during training, frequently resulting in faster convergence and better efficiency than basic optimisers such as SGD. The depth of a neural network is affected by the number of hidden layers. It can affect the ability to detect complex patterns in the data. Adding more hidden layers let the model to find complex patterns which might reduce loss.

However, the deeper networks are more prone to overfitting and need more careful tuning of other parameters. For learning rate, it controls the step size during optimization. High learning rate can speed up training but can cause instability and on the other hand low rate stay stable convergence but will slow down the training. And lastly, batch size affects how the model changes its weight. Smaller batches will be noisier but more frequent updates to help avoid local minima. while larger batches give more stable changes but can lead to overfitting happen. In conclusion, balancing the above factors (epochs, optimizers, hidden layers, learning rate, and batch size) is important for optimizing neural network performance. The goal is to minimize loss while ensuring the model generalizes well to unseen data.

Standard scaling is a preprocessing technique commonly utilised in neural networks, which is an operation to rescale the features of the data to have a result where mean value is 0 and standard deviation is 1. This aids to ensure the contribution of each feature to the learning process will be equal. Standard scaling leads to improved convergence where it prevents features with large ranges from dominating the learning process, resulting in a better performance as the weights of each feature is more uniform during the process.

Training time (seconds, s)					
Trial1	Trial2	Trial3	Trial4	Trial5	Average
8.0621	8.1804	8.0874	8.3183	8.0389	8.1374

Table 2.4a: Training time for Neural Network method (epochs = 50, batch size = 1).

Prediction time (seconds, s)					
Trial1	Trial2	Trial3	Trial4	Trial5	Average
0.0822	0.0893	0.0886	0.0844	0.1135	0.0916

Table 2.4b: Prediction time for Neural Network method (epochs = 50, batch size = 1).

	R^2 squared	MAE	MSLE	RMSE
Trial1	0.9106	106.7507	0.0750	142.9200
Trial2	0.8928	108.6271	0.0813	156.5464
Trial3	0.9130	104.4005	0.0813	141.0331
Average	**0.9055**	**106.5928**	**0.0792**	**146.8332**

Table 2.4c: Performance of Neural Network method (epochs = 50, batch size = 1, learning rate = 0.001, Optimizer = Adam). (Default)

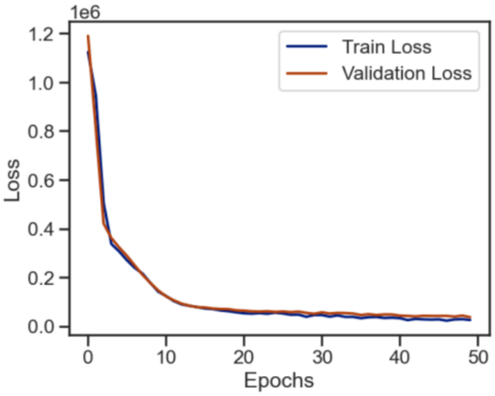


Figure 2.4c: Pattern of train loss and validation loss in neural network for Table 2.4c.

	R^2 squared	MAE	MSLE	RMSE
Trial1	0.9317	83.2788	0.0684	124.9413
Trial2	0.9443	82.7161	0.0563	112.8646
Trial3	0.9161	87.6258	0.0702	138.4645
Average	**0.9307**	**84.5402**	**0.0650**	**125.4235**

Table 2.4d: Performance of Neural Network method (**epochs = 100**, batch size = 1, learning rate = 0.001, Optimizer = Adam).

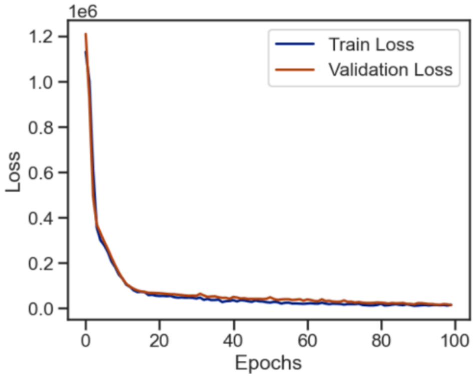


Figure 2.4d: Pattern of train loss and validation loss in neural network for Table 2.4d.

	R^2 squared	MAE	MSLE	RMSE
Trial1	0.8973	96.5538	0.08104	153.2308
Trial2	0.9271	106.1377	0.09357	129.1165
Trial3	0.9258	103.7193	0.07445	130.2236
Average	**0.9167**	**102.1369**	**0.08302**	**137.5236**

Table 2.4e: Performance of Neural Network method (epochs = 50, batch size = 1, **learning rate = 0.01**, Optimizer = Adam).

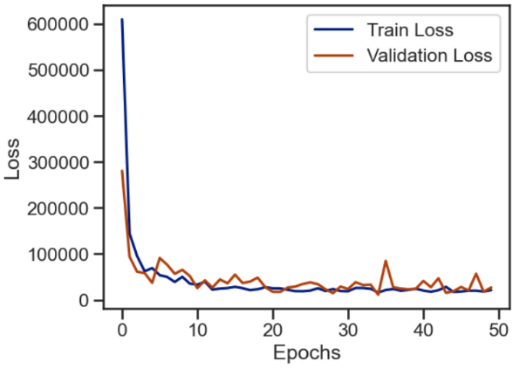


Figure 2.4e: Pattern of train loss and validation loss in neural network for Table 2.4e.

	R^2 squared	MAE	MSLE	RMSE
Trial1	0.6215	228.6319	0.3286	294.1224
Trial2	0.5908	244.2595	0.3523	305.8303
Trial3	0.5693	252.0670	0.3768	313.7540
Average	**0.5939**	**241.6528**	**0.3526**	**304.5689**

Table 2.4f: Performance of Neural Network method (epochs = 50, **batch size = 10**, learning rate = 0.001, Optimizer = Adam).

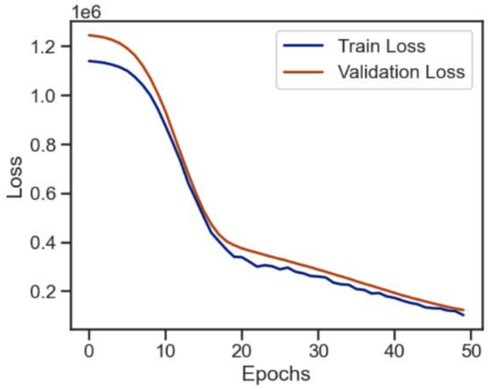


Figure 2.4f: Pattern of train loss and validation loss in neural network for Table 2.4f.

	R^2 squared	MAE	MSLE	RMSE
Trial1	0.9393	83.6913	0.0430	117.7515
Trial2	0.9401	93.8043	0.0751	116.9628
Trial3	0.9222	98.8627	0.0547	133.3734
Average	**0.9334**	**92.1194**	**0.0576**	**122.6959**

Table 2.4g: Performance of Neural Network method (epochs = 50, batch size = 1, **learning rate = 0.000001**, **Optimizer =SGD**).

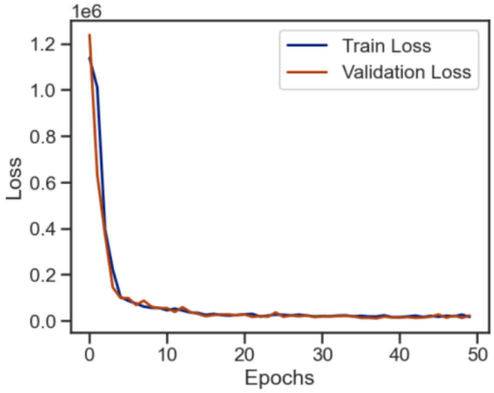


Figure 2.4g: Pattern of train loss and validation loss in neural network for Table 2.4g.

Increasing the number of epochs in the neural network to 100 significantly improves model performance. The R^2 value rises to 0.9307, indicating enhanced explanatory power. Moreover, the decreases in MSLE, MAE, and RMSE suggest that the model benefits from more iterations. On the other hand, increasing the learning rate to 0.01 with the Adam optimizer has mixed effects: the R^2 value remains high, but MAE and MSLE slightly increase while RMSE slightly decreases compared to the default settings. This suggests that a higher learning rate can accelerate convergence but may also introduce instability, potentially leading to less accurate predictions.

Additionally, increasing the batch size to 10 results in a notable drop in performance. The R^2 value falls to 0.5939, indicating that the model explains only 59.39% of the variance, and all error metrics increase significantly. This is likely because less frequent weight updates with a larger batch size hinder the model's ability to adapt efficiently. Lastly, when using the SGD optimizer instead of Adam, a much lower learning rate is required due to its greater instability. A significant reduction in the learning rate results in an R^2 value of 0.9334, along with lower error metrics, particularly in MAE and MSLE. This suggests that finer weight updates can yield more precise results.

Comparison of performance between the algorithm methods

Aspect Method	R^2 squared	MAE	MSLE	RMSE	Training time (s)	Prediction time (s)
Random Forest	0.9955	24.7386	0.0017	36.3594	0.1997	0.0084
Linear Regression	0.9863	41.4595	0.0089	58.2659	0.0015	0.0003
KNN	0.9889	36.6030	0.0048	52.6269	0.0046	0.0074
Neural network (default)	0.9055	106.5928	0.0792	146.8332	8.1374	0.0916

(epochs = 50, batch size = 1, learning rate = 0.001, Optimizer = Adam). (Default)

From the above table of comparison, random forest algorithm will be the best model for predicting the number of Covid ICU patients for the 30 input features. The results are shown in the table, random forest has the highest R-squared value and lowest Mean square logarithm error. High R-square value means higher fitness of model to predict the target while lowest MSLE means smaller difference between the prediction value and truth value. On the other hand, the training time and prediction time of random forest is not the shortest among the 4 algorithms. However, it produces a more accurate prediction and hence random forest is still preferable.

References

1. Singh, P.V. (2023). *All About Min-max scaling*. [online] Medium. Available at: https://medium.com/@poojaviveksingh/all-about-min-max-scaling-c7da4e0044c5. [Accessed 25 August 2024].
2. Minitab.com. (2024). *R-squared vs number of trees plot for Random Forests® Regression*. [online] Available at: https://support.minitab.com/en-us/minitab/help-and-how-to/statistical-modeling/predictive-analytics/how-to/random-forests-regression/interpret-the-results/r-squared-vs-number-of-trees-plot/#:~:text=The%20R%2Dsquared%20vs%20Number. [Accessed 25 August 2024].
3. Built In. (n.d.). *Random Forest: A Complete Guide for Machine Learning*. [online] Available at: https://builtin.com/data-science/random-forest-algorithm#:~:text=Random%20forest%20is%20an%20algorithm [Accessed 26 August 2024].
4. Amazon Web Services, Inc. (n.d.). *What is Linear Regression? - Linear Regression - AWS*. [online] Available at: https://aws.amazon.com/what-is/linear-regression/#:~:text=Linear%20regression%20is%20a%20data. [Accessed 26 August 2024].
5. IBM (2024). *What Is Linear Regression? | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/linear-regression#:~:text=IBM-. [Accessed 26 August 2024].Indeed Career Guide. (n.d.). *Linear vs. Logistic Regression (Differences and Limitations)*. [online] Available at: https://ca.indeed.com/career-advice/career-development/linear-vs-logistic-regression.
6. Joby, A. (2023) *K Nearest neighbor or KNN algorithm and it's essence in ML*. [online] Available at: https://learn.g2.com/k-nearest-neighbor. [Accessed: 25 August 2024]
7. Coşgun, H.H. (2023) *Which data scaling technique should I use ?* Medium. [online] Available at: https://medium.com/@hhuseyincosgun/which-data-scaling-technique-should-i-use-a1615292061e#:~:text=StandardScaler%20is%20a%20data%20preprocessing,assume%20a%20standard%20normal%20distribution. [Accessed: 26 August 2024]
8. *IEEE Xplore Full-Text PDF:* (n.d.). https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9318195

9. *Importance of epoch, batch size, iterations in deep learning and AI training*. (2024, August 9). https://www.sabrepc.com/blog/Deep-Learning-and-AI/Epochs-Batch-Size-Iterations