ISM 6136- Data Mining and Predictive Analytics

Main Project – Supervised Machine Learning

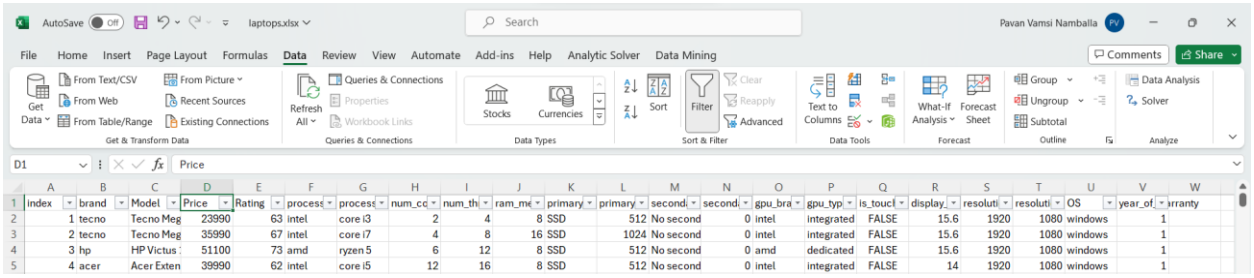
This is the project work done by **Pavan Vamsi Namballa(Z23752169)** and **Sai Sahith Nallakuntla(Z23748226)**, for the course **ISM6136 – Data Mining/Predictive Analytics**. The dataset that has been selected for this project is from **Kaggle.com**. It is available on the site as **Brand Laptops Dataset**. The author is **Bhavik Jikadara**.

Link: - https://www.kaggle.com/datasets/bhavikjikadara/brand-laptops-dataset

The dataset initially contains 992 records, where each of the records represents a single laptop. It has up to 22 fields among which the field **Price** is the output variable, which takes up continuous values. The remaining 21 fields act as predictor variables, based on which the value of price depends upon.

Our goal is to build a **Supervised Machine Learning model** which can predict the value for the **price** whenever a new observation is found. Since the output values are numerical and continuous, we look up to the **Regression** algorithms.

But before performing any regression test, we must make sure that the data is cleaned up. To do so, we need to analyze all the fields present in the dataset.



index	brand	Model	Price	Rating	process	process	num_co	num_th	ram_me	primary	primary	second	second	gpu_bra	gpu_typ	is_touc	display_	resoluti	resoluti	OS	year_of	rranty
1	tecno	Tecno Meg	23990	63	intel	core i3	2	4	8	SSD	512	No second	0	intel	integrated	FALSE	15.6	1920	1080	windows	1	
2	tecno	Tecno Meg	35990	67	intel	core i7	4	8	16	SSD	1024	No second	0	intel	integrated	FALSE	15.6	1920	1080	windows	1	
3	hp	HP Victus	51100	73	amd	ryzen 5	6	12	8	SSD	512	No second	0	amd	dedicated	FALSE	15.6	1920	1080	windows	1	
4	acer	Acer Exten	39990	62	intel	core i5	12	16	8	SSD	512	No second	0	intel	integrated	FALSE	14	1920	1080	windows	1	

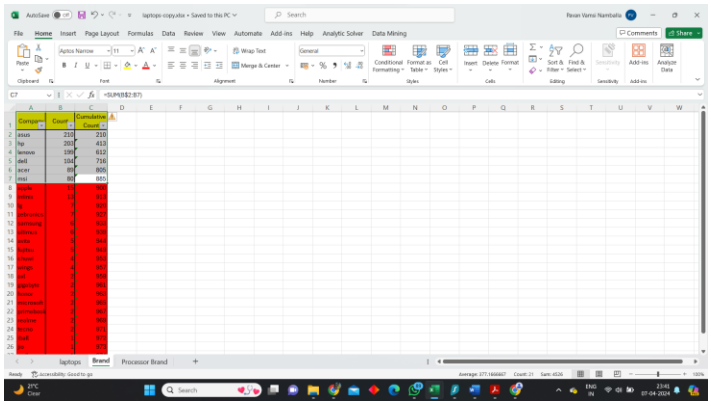
Let us Discuss each of the fields in detail so that we can decide on what data to be deleted or reduced.

1. **Index-**

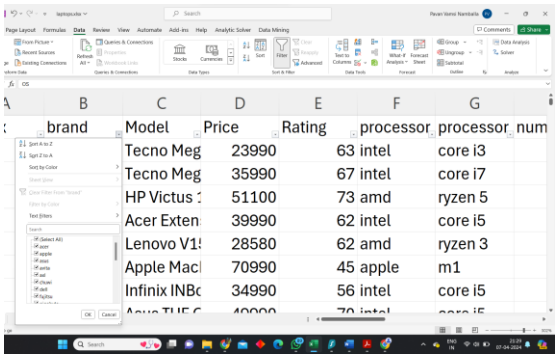
   The unique identifier of every laptop in the dataset. It does not make any difference in computation of the output. Hence, can be **ignored**.

2. **Brand-**

   The company that has manufactured the laptop being represented in the record. It is a field that contains discrete data and had more than 10 values. We reduced it to 6.

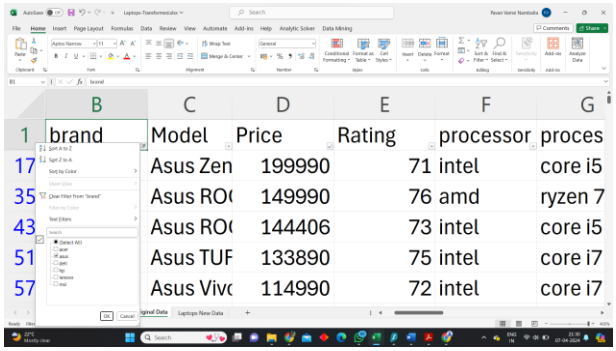


Before Transformation: -



brand	Model	Price	Rating	processor	processor_	num
	Tecno Meg	23990	63	intel	core i3	
	Tecno Meg	35990	67	intel	core i7	
	HP Victus	51100	73	amd	ryzen 5	
	Acer Exten	39990	62	intel	core i5	
	Lenovo V15	28580	62	amd	ryzen 3	
	Apple Mac	70990	45	apple	m1	
	Infinix INBc	34990	56	intel	core i5	

After Transformation: -



	brand	Model	Price	Rating	processor	proces
17		Asus Zen	199990	71	intel	core i5
35		Asus ROG	149990	76	amd	ryzen 7
43		Asus ROG	144406	73	intel	core i5
51		Asus TUF	133890	75	intel	core i7
57		Asus Vivo	114990	72	intel	core i7

3. Model

Unique Name assigned by the manufacturer. Every record in the dataset takes up unique textual values. They have zero significance in determining the output. Hence, they can be ignored as a predictor variable.

4. Price

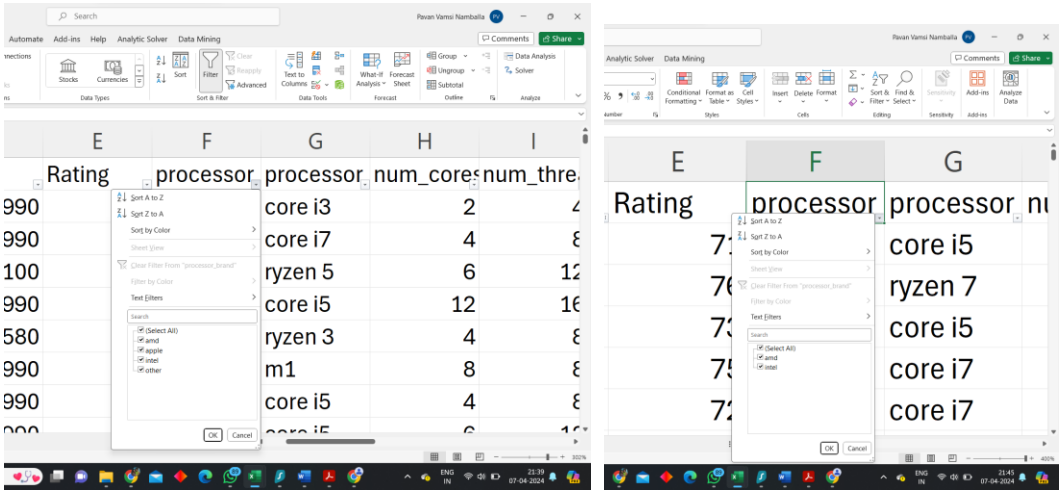
This is the output variable whose value is determined by the remaining variables. Our goal is to build a suitable model that predicts the value of this field.

5. Rating

A Numeric value given to the model by the users who have used it previously. Having a higher rating means having higher reliability.

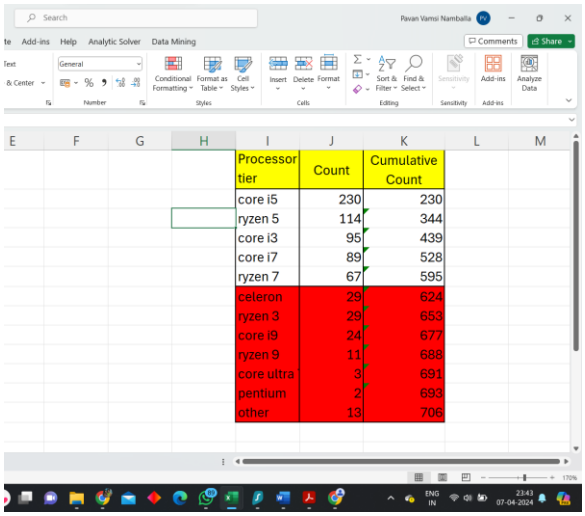
6. Processor_brand

Name of the company who manufactured the processor present in the laptop. In the original data, we have 4 discrete values: **amd, apple, intel** and **other**. Since it has only a lesser number of values, we ignored making changes in the field data. However, upon further preprocessing done based on remaining fields, we are left only with two of the values: **amd** and **intel**.

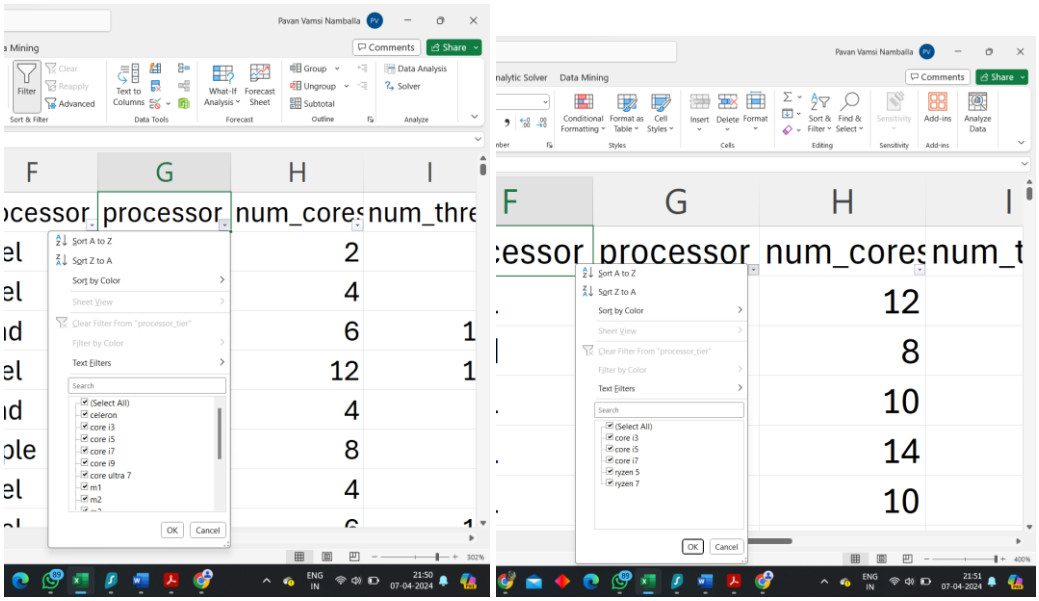


7. Processor_tier

The name given to the processor by the manufacturer based on the performance, capability, and other features.



It initially had 11 discrete values, but upon transformation, it has been reduced to 5.



8. Num_cores

Cores refers to individual physical processing units within a computer chip. Having a greater number of cores paves the path for efficient parallel preprocessing. It takes up continuous values.

9. Num_threads

A thread is the smallest unit of execution that can be handled by the processor. A larger task is subdivided into smaller threads which helps in the speeding up of execution. The property **Multithreading** is one of the greatest selling points a processor can possess. In our dataset, it takes up continuous values.

10. Ram_memory

RAM stands for **Random Access Memory**. It is an example of primary storage of a computer. It is a highly volatile memory whose contents get erased when the power is turned off. It is used by the processor and the computer for easy access of data. Its size would usually be represented in exponents of 2.

Example: - 2^2, 2^3, 2^4, 2^5... etc.

We have it in the form of numeric discrete data. We have values up to 7. They are minimized to four values up on preprocessing.

11. Primary_storage_type

Primary Storage usually comes in two forms: SSD (Solid-State Drive) and HDD (Hard-Disk Drive). Hence it is a discrete variable with two values. But, upon transformation based on the remaining fields, we are left only with the SSD. Hence, it can be **removed**.

12. Primary_storage_capacity

It is same as **ram_memory** but is used in a broader sense. It takes more than 7 discrete values, but upon preprocessing, they are reduced to 4.

13. secondary_storage_type

Secondary Storage is the memory unit where the data is stored permanently. In our dataset it takes up to two discrete values. Before the transformation based on fields 1 to 12, it had two values, but on transformation, it took up unary values. Hence, it is **removed**.

14. secondary_storage_capacity

Just like primary_storage_capacity, it takes up discrete values which are exponents of 7. Upon transformation, it has started taking up unary values. Hence, it has also been **removed**.

15. gpu_brand

The company which manufactured the GPU. GPU stands for Graphics Processing Unit. It takes up textual categorical values.

16. gpu_type

Just like gpu_brand, it also takes up textual categorical(discrete) data. Before the preprocessing, it was 3, and after preprocessing, it takes 2 values.

17. is_touch_screen

It is a Boolean variable (**True/False**) which determines if the laptop has touch screen feature or not.

18. display_size

Display Size of a laptop refers to the physical dimensions of the screen, usually measured in inches. It is the distance between two opposite corners of the screen. In our dataset, it takes up continuous values.

19. resolution_width

Resolution width is the horizontal component of display resolution, commonly expressed in pixels. It signifies the quantity of pixels arranged horizontally on the screen. For instance, in a resolution specification such as "1920 x 1080," the resolution width would be 1920 pixels.

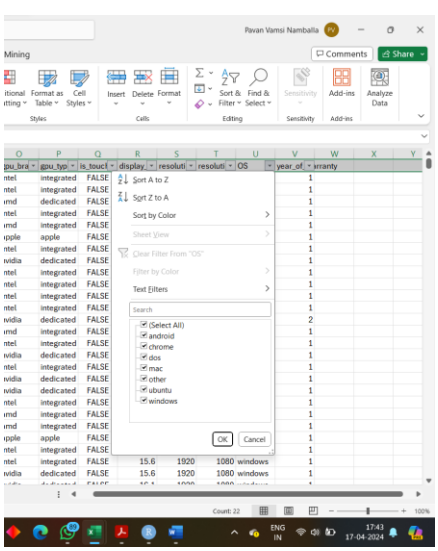
20. resolution_height

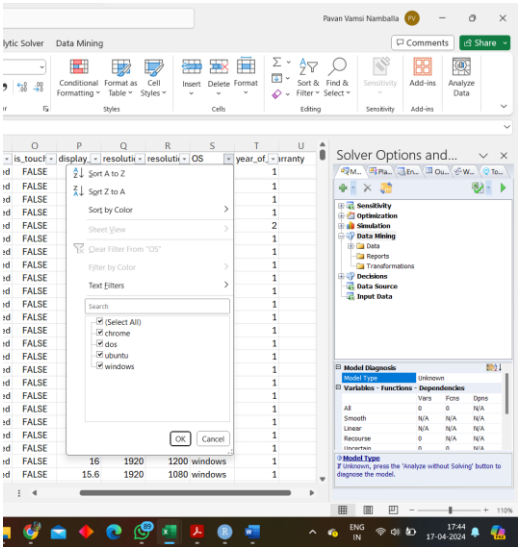
Resolution width is the vertical component of display resolution. It signifies the quantity of pixels arranged vertically on the screen. For instance, in a resolution specification such as "1920 x 1080," the resolution width would be 1920 pixels. In our dataset, it takes up continuous values.

21. OS

“An operating system is the most important software that runs on a computer. It manages the computer's memory and processes, as well as all its software and hardware. It also allows you to communicate with the computer without knowing how to speak the computer's language. Without an operating system, a computer is useless.”(Definition from https://edu.gcfglobal.org/en/computerbasics/understanding-operating-systems/1/).

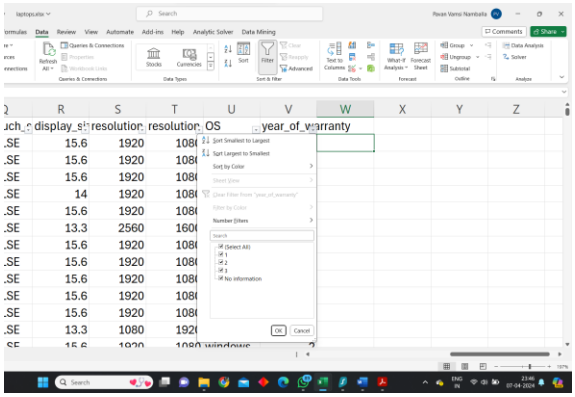
Initially, it took 7 discrete values. We reduced it to 4.





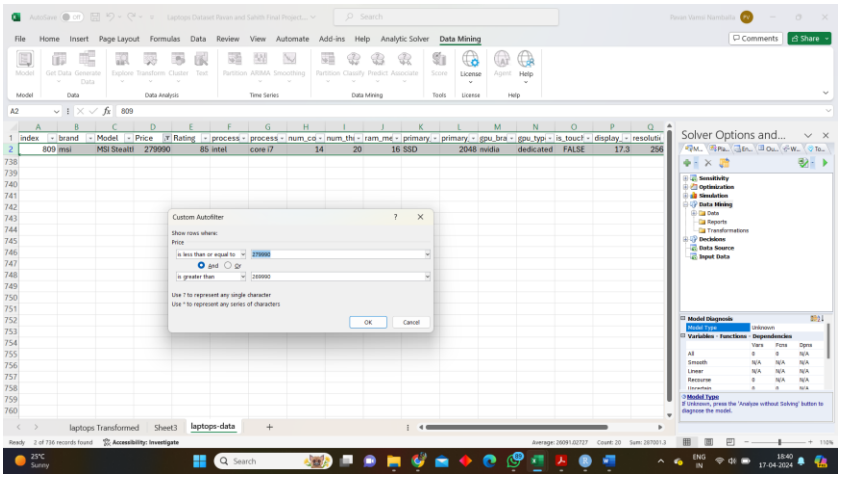
22. Year of Warranty

Time period after the purchase, where the seller guarantees the customer about the quality and performance of the laptop. It takes up values: 1, 2 and 3. But there are rows where the value for this field is given as **No Information**. There are 19 rows with that value. Upon deleting them, we have many values in the fields from A-U that got deleted.

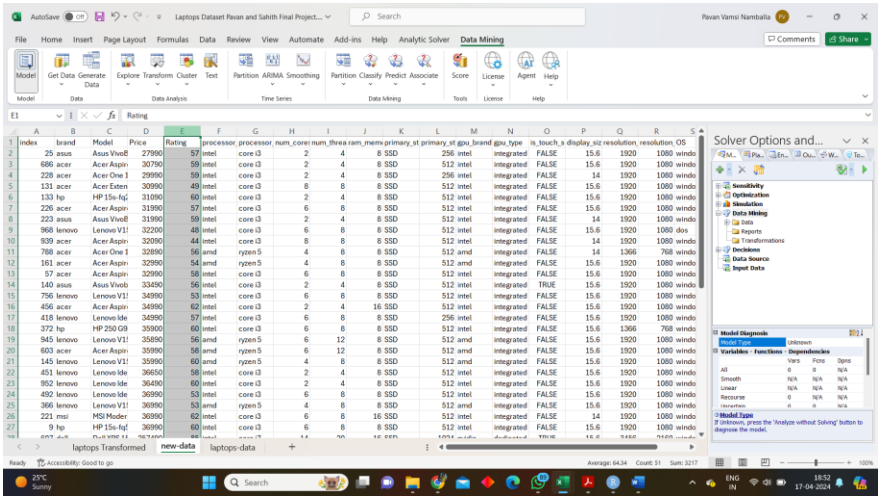


After doing all the transformations, we are now left with 762 records. The Final transformed Sheet is named "**Laptops Dataset Pavan and Sahith Final Project.xlsx**", and the transformed sheet is "**laptops-data**". We create a new sheet called "**new data**", where arbitrarily chosen records from the "**laptops-data**" are pasted up to a count of 57. Predictions are supposed to be done on this data by **scoring**.

The records in the dataset after the transformation have a price range of **27990-279990**.To select the records, "**laptops-data"** sheet is opened, and a custom filter is applied based on every price range possible. Arbitrarily chosen records of count 57 are copied into the **"new data"** sheet.



Upon doing so, the **laptops-data** sheet has a row count of 705 while the **new-data** sheet has a row count of 57.



We are now done with the data reduction and separation. It is the time, the machine learning algorithms are applied on the data.

We employ two popular Supervised Machine Learning Algorithms: -

1. Linear Regression
2. Regression Trees

Linear Regression

Linear Regression is a **Supervised Machine Learning** Regression Algorithm where the model is in the form of a straight line.

Let us suppose that the given predictors are [x_0, x_1, x_2, x_3,……., x_n], and the output is Y.

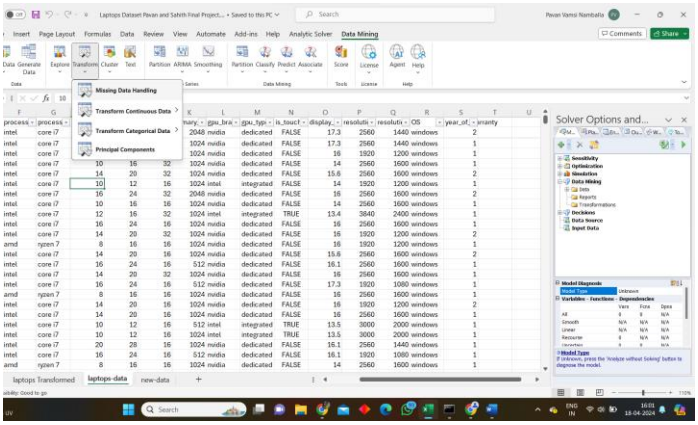
The output follows the below equation: -

$$Y = W^T X = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + \cdots \ldots \ldots \ldots + w_n x_n$$

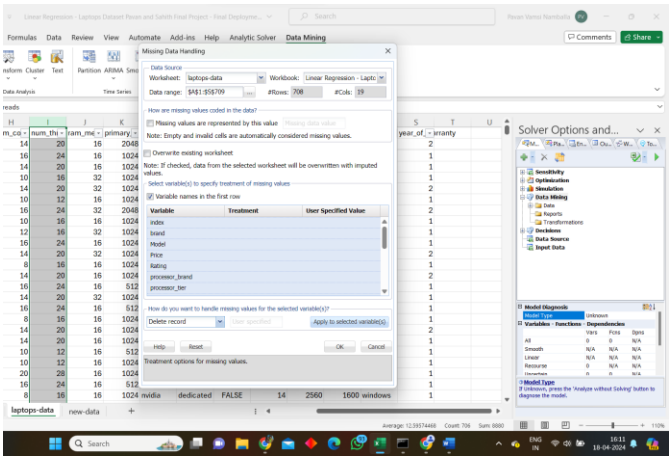
We have the weights [w_0, w_1,w_2,w_3,……., w_n], whose values are to be found based on the computations done by the algorithm. The algorithm is all about finding the appropriate values of the weights, based on which correct line is found where we have the least errors. We have functions like the SSE (Total Sum of Squared Errors), RMSE (Root Mean Squared Error) whose value is to be minimized. The data is divided into **Training** and **Validation** sets. The algorithm is applied on the **training set**, where the model is obtained, which is in the form of a straight line. Two or more models are created and the one with highest value for the Regression Coefficient, **R^2** and least **RMSE**, is considered the best. The models should ideally have a validation R^2 value of at least 70%.

1. Clean Up the Data

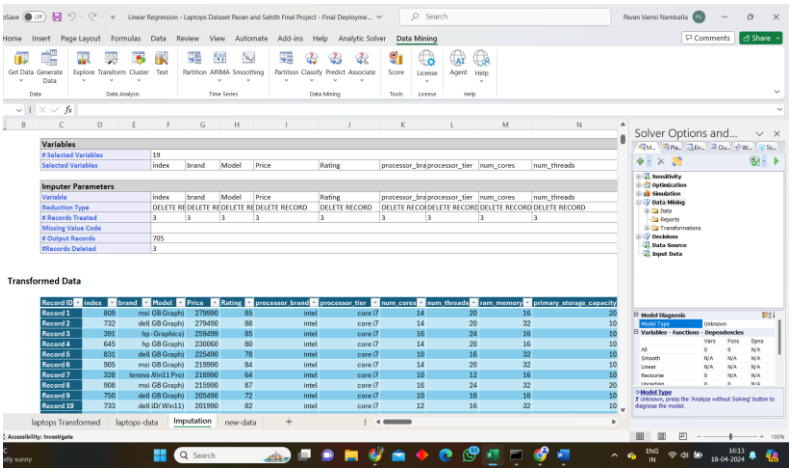
The first and foremost step is to make sure that there are no missing values present in the dataset.



Click anywhere on the data and then go to the **Data Mining** tab.
Go to **Transform -> Missing Data Handling**.



On clicking on the **Missing Data Handling** button, we are navigated to a dialog box with the name **Missing Data Handling**. Make sure that the **Data** Range under the **Data Source** is set to the correct value (In our case, it is 709). Select all the column names and set the treatment to **Delete Record**. Click on **Apply to Selected Variables** and then select **OK.**



We can see a new sheet created with the name **Imputation**, where we have 3 rows with empty records deleted and we have a count of 705.

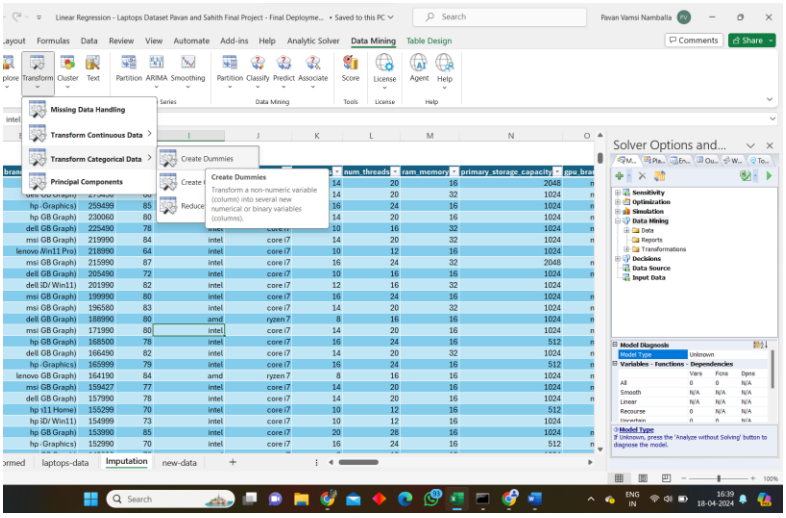
Upon making sure that no missing values are present, it is the time to make sure that there are no predictors present which have textual or discrete data. The best solution is to create dummies for them.

We have the following fields which take up textual values: **brand, model, processor_brand, processor_tier, gpu_brand, gpu_type** and **OS**.

Among them, the variable **model** takes unique textual values and doesn't have any significance in determining the output. So, we do not consider that for the process of dummy creation.

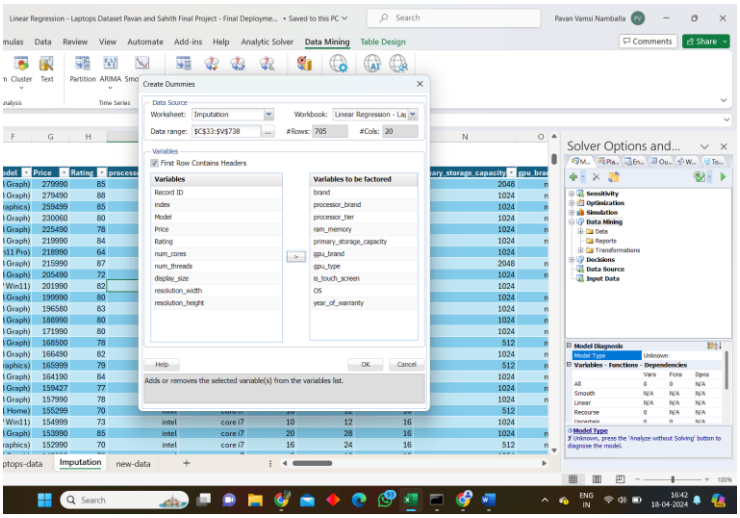
Among the numerical fields, we have **ram_memory, primary_storage_capacity, is_touch_screen** and **year_of_warranty** which takes discrete values.

Hence, we create dummies for the following variables: - **model, processor_brand, processor_tier, gpu_brand, gpu_type**, **OS, ram_memory, primary_storage_capacity, is_touch_screen** and **year_of_warranty**.



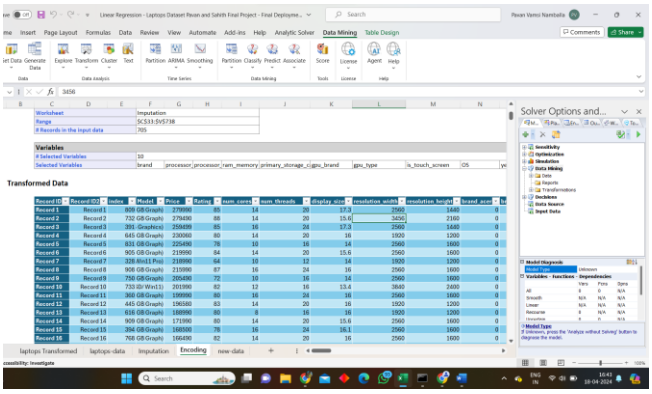
Click anywhere on the data and go to the **Data Mining** tab.

Go to **Transform -> Transform Categorical Data -> Create Dummies**.



Select the appropriate variables and click on **OK**.

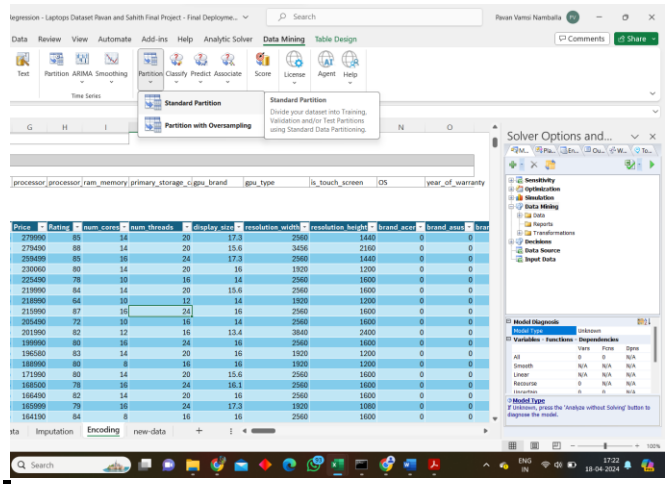
Upon doing so, we have a new sheet called **encoding**, where we have the dummies created. It is on this sheet, where we perform the partition.



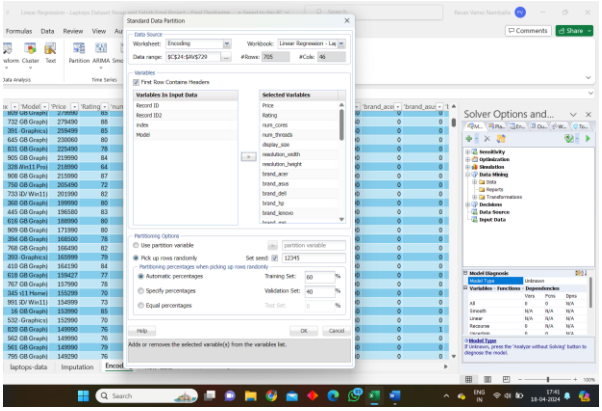
2. Time for the Partition

In this step, we divide the data into training and validation sets.
Click anywhere on the **encoding** sheet and go to the **Data Mining** tab.
Click on the **Partition** dropdown which is located left to the **Classify** dropdown.

Now click on **Standard Partition**.



The variables **index** and **model** which has no significance in determining the output are ignored along with the dynamically assigned variables **Record ID** and **Record ID2**.

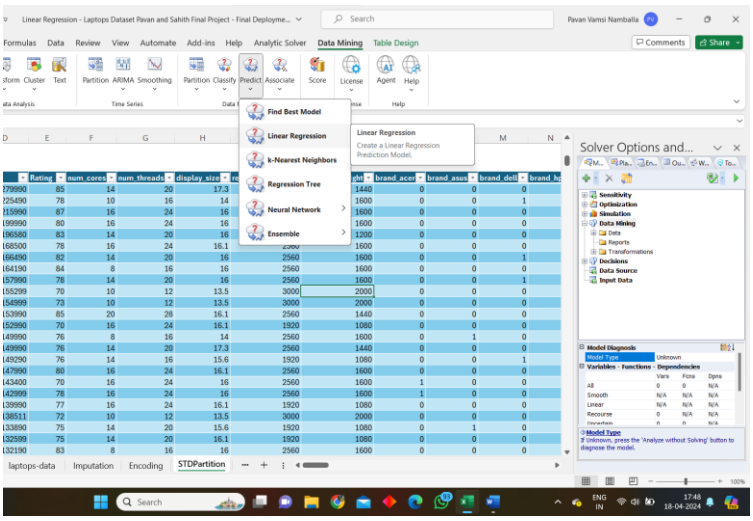


The ratio of training to validation data is **automatically set** to 60:40. To set manually, select the radio button **Specify Percentages**.

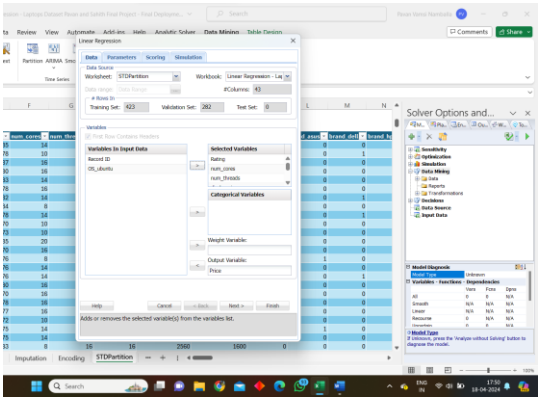
We have a new sheet created with the name **STDPartition**. It is on this sheet, where the algorithm is applied on.

3. Apply the Algorithm

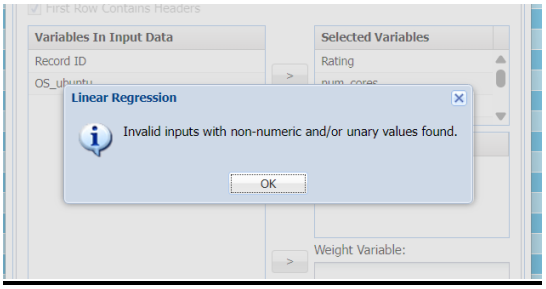
Click anywhere on the **STDPartition** sheet and go to the **Data Mining** tab.
Go to **Predict -> Linear Regression**.



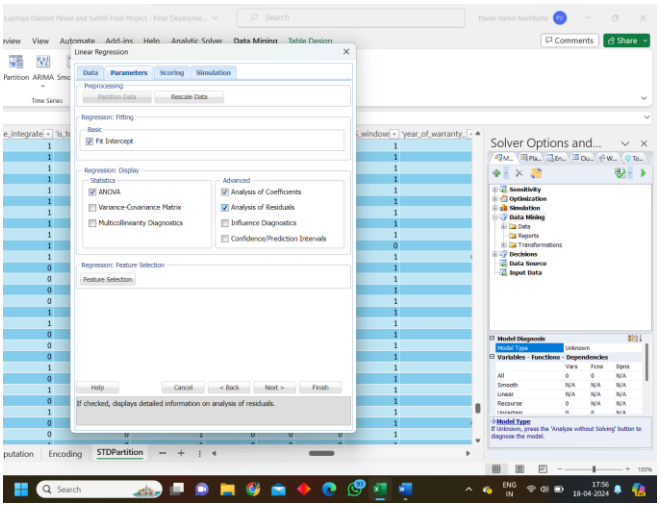
The dynamically assigned variable **Record ID** is ignored. The **Price** variable is made the output variable. The remaining ones are made the input variables.



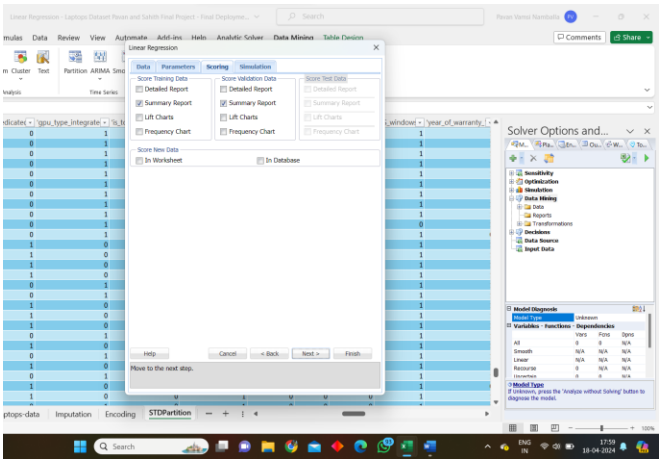
The variable **US_ubuntu** contains unary values. Hence, it is ignored by the algorithm.



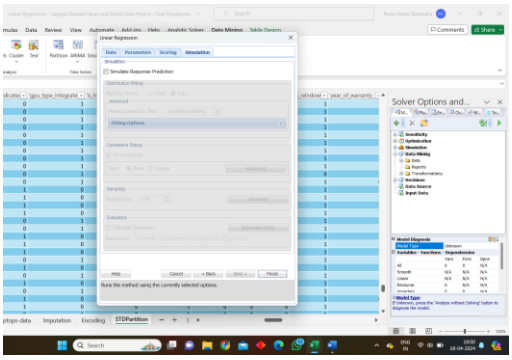
Upon selecting the data, in **Linear Regression** window, under the **parameters** tab, make sure that the checkboxes **ANOVA**, **Analysis of Coefficients** and **Analysis of Residuals** are ticked and click on **Next**.



Under **Scoring**, check **Summary Report** for training as well as validation data and click on **Next**.



Under **Simulation**, click on **Finish**.



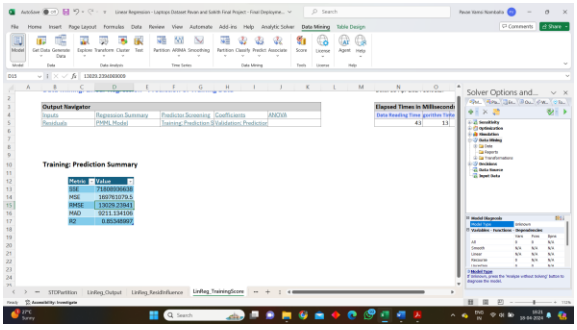
Upon doing so, the algorithm is applied, and we have 5 new sheets created with the names **LinReg_output**, **LinReg_ResidInfluence**, **LinReg_TrainingScore**, **LinReg_**ValidationScore and **LinReg_Scored**.

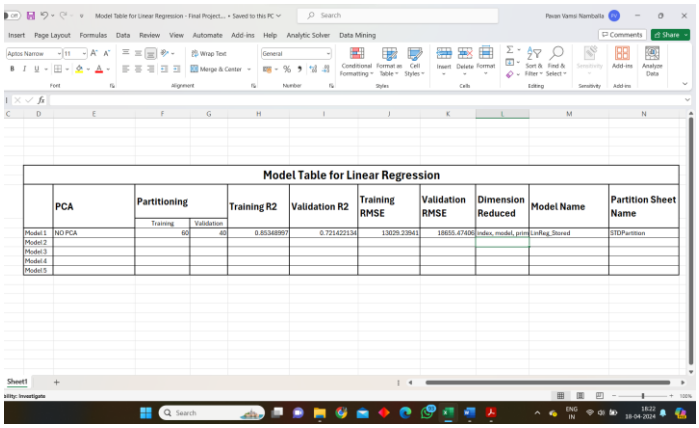
LinReg_Scored is the name of the model that has been created.

4. Evaluate the Model

We have now created a new workbook with the title "**Model Table for Linear Regression - Final Project.xlsx**" under which we note down the metrics of the model created.

From **LinReg_TrainingScore**, we noted down the values of **Training R2** and **Training RMSE** and from **LinReg_ValidationScore**, we noted down the values of **Validation R2** and **Validation RMSE**.





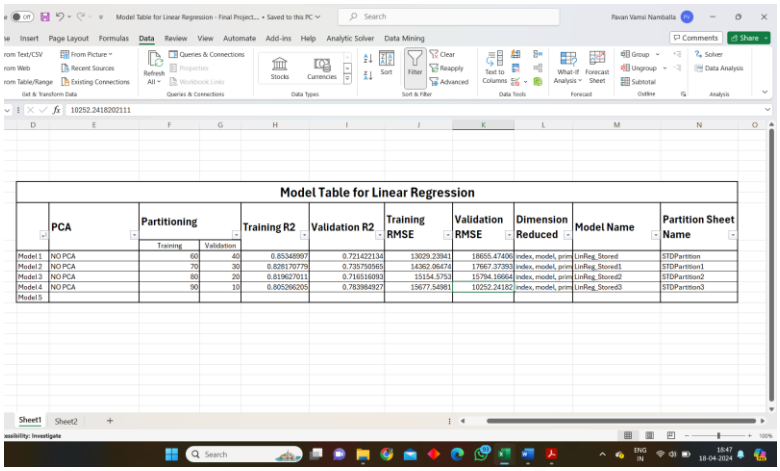
Model Table for Linear Regression

	PCA	Partitioning		Training R2	Validation R2	Training RMSE	Validation RMSE	Dimension Reduced	Model Name	Partition Sheet Name
		Training	Validation							
Model 1	NO PCA	60	40	0.85348997	0.721422134	13029.23941	18655.47406	index, model, prim	LinReg_Stored	STDPartition
Model 2										
Model 3										
Model 4										
Model 5										

We used the same table to evaluate the models that are going to be created further.

5. Create More Models

Using the steps from 1-4, we create three more models and evaluate them using the same metrics and note them down in the same **Model Table**.



Model Table for Linear Regression

	PCA	Partitioning		Training R2	Validation R2	Training RMSE	Validation RMSE	Dimension Reduced	Model Name	Partition Sheet Name
		Training	Validation							
Model 1	NO PCA	60	40	0.85348997	0.721422134	13029.23941	18655.47406	index, model, prim	LinReg_Stored	STDPartition
Model 2	NO PCA	70	30	0.828170779	0.735750565	14362.06474	17667.37393	index, model, prim	LinReg_Stored1	STDPartition1
Model 3	NO PCA	80	20	0.819627011	0.716516093	15154.5753	15794.16664	index, model, prim	LinReg_Stored2	STDPartition2
Model 4	NO PCA	90	10	0.805296205	0.783984927	15677.54981	10252.24182	index, model, prim	LinReg_Stored3	STDPartition3
Model 5										

In this way, we have the first 4 models created and evaluated.

But for the fifth model, we perform the **Principal Component Analysis** just before the partition (Step 2). It is performed to identify the topmost predictors. It makes use of data statistics, **covariance (or) correlation** between every possible pair of two fields (excluding the output fields). As correlation does not really show the variance, correlation between the variables is considered as a parameter. It is one of the best **Dimensionality Reduction** algorithms.

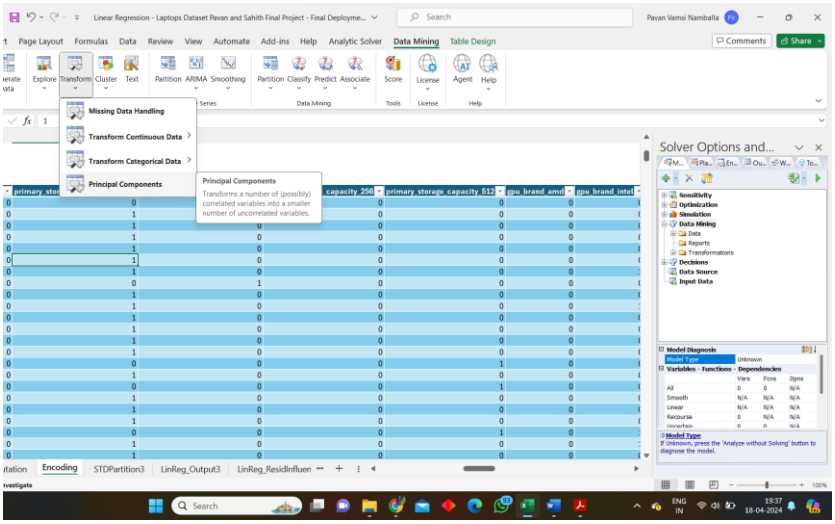
$$Covariance(x,y) = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{n}$$

$$Correlation(x,y) = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{\sqrt{\Sigma(x-\bar{x})^2\Sigma(y-\bar{y})^2}}$$

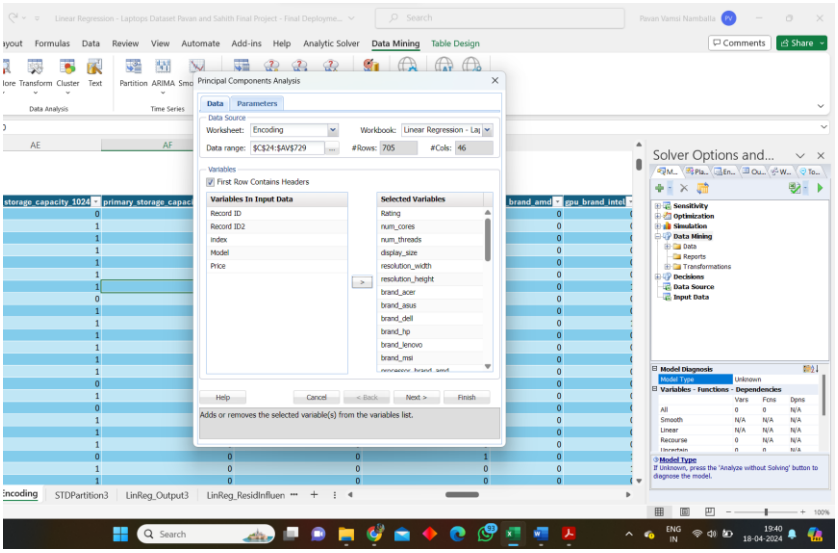
PCA tries to eliminate the ones that are very highly correlated and keeps the ones that are really capturing the changes in the dataset. A Larger number of correlated variables are transformed into a smaller number of uncorrelated variables. Overall, the PCA is a technique that emphasizes variation and brings out strong patterns in the dataset.

To do that, click anywhere on the **encoding** tab and go to the **Data Mining** tab.

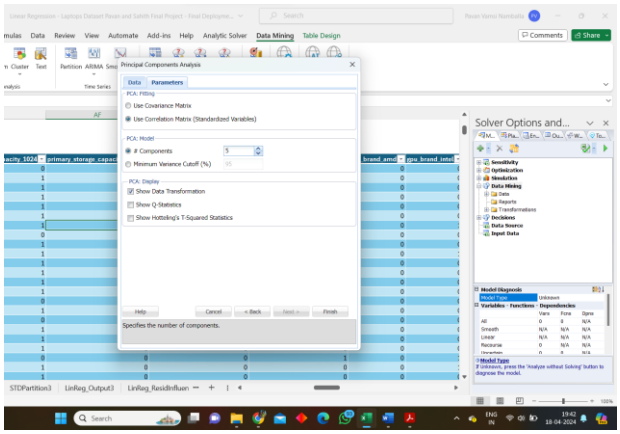
Go to **Transform -> Principal Components.**



In principal Component analysis, we include only the predictor variables and exclude the output variable.

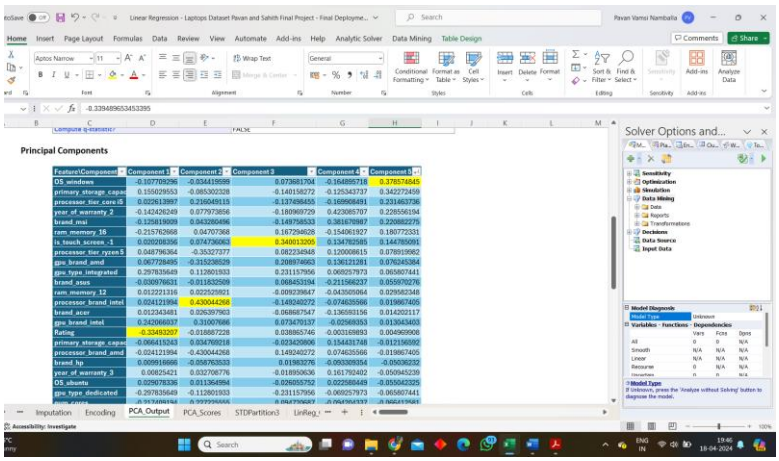


We selected the “Top 5 Components” by selecting the number of components as 5, under the **Parameters** tab. Click on **Finish**.

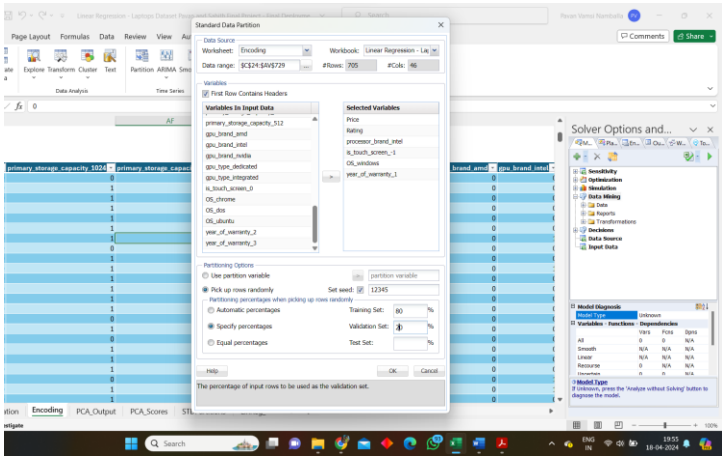


We have a sheet called **PCA_Output** where we get the top 5 components.

Top 5 Components are: - **Rating**, **Processor_brand_intel**, **is_touch_screen_-1**, **year_of_warranty_1** and **OS_windows**.

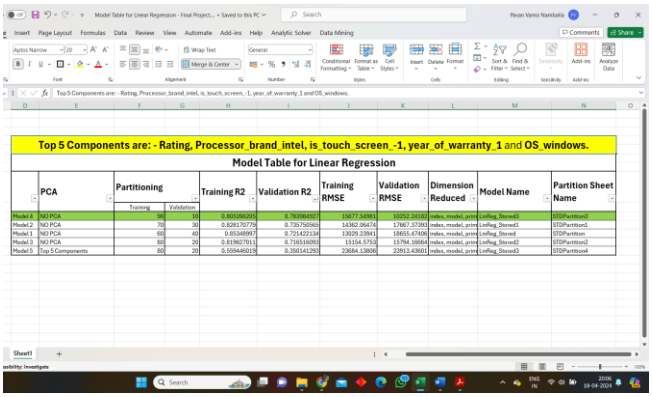


During the partition, only the top 5 components are included along with the output variable **Price** with a partition ratio of 80:20. We work on steps 2 to 4 and evaluate the model.



We can see that model #4 "**LinReg_stored3**" is the best model as it has highest validation accuracy and lowest RMSE among all the ones created.

For this dataset, PCA does not work as it yielded accuracies less than 70%.

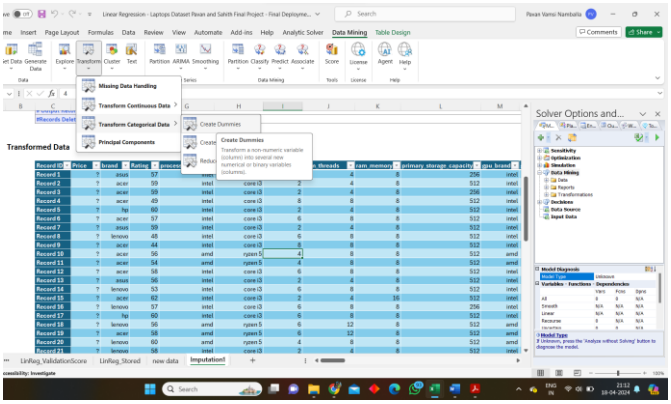


6. Deploy the best Model!

It is the time we deploy the model, meaning, the best model is applied on the **new data** to make predictions.

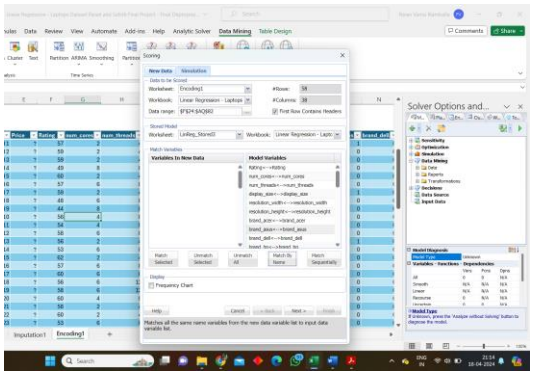
We deleted the unnecessary variables like **index, model** which are not predictors.

The prerequisite for any model deployment is that the data should be clean and there should not be any missing values present. We create dummies for discrete data, just the way we did for the original data. Extra records are added from the original dataset to make sure that all variables from the model are present in the new data.

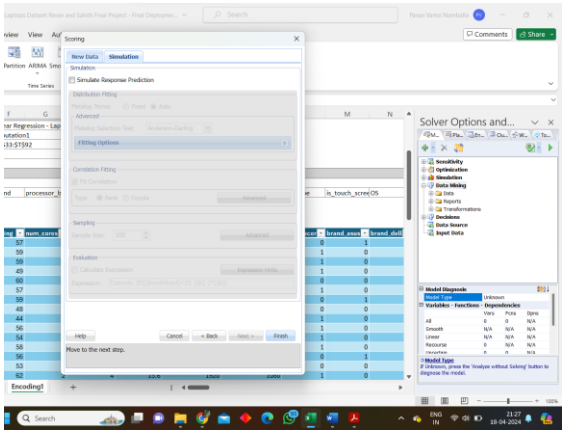


Click anywhere on the data and open the **Data Mining** tab and perform missing data handling. Upon doing so, create dummies for the discrete data.

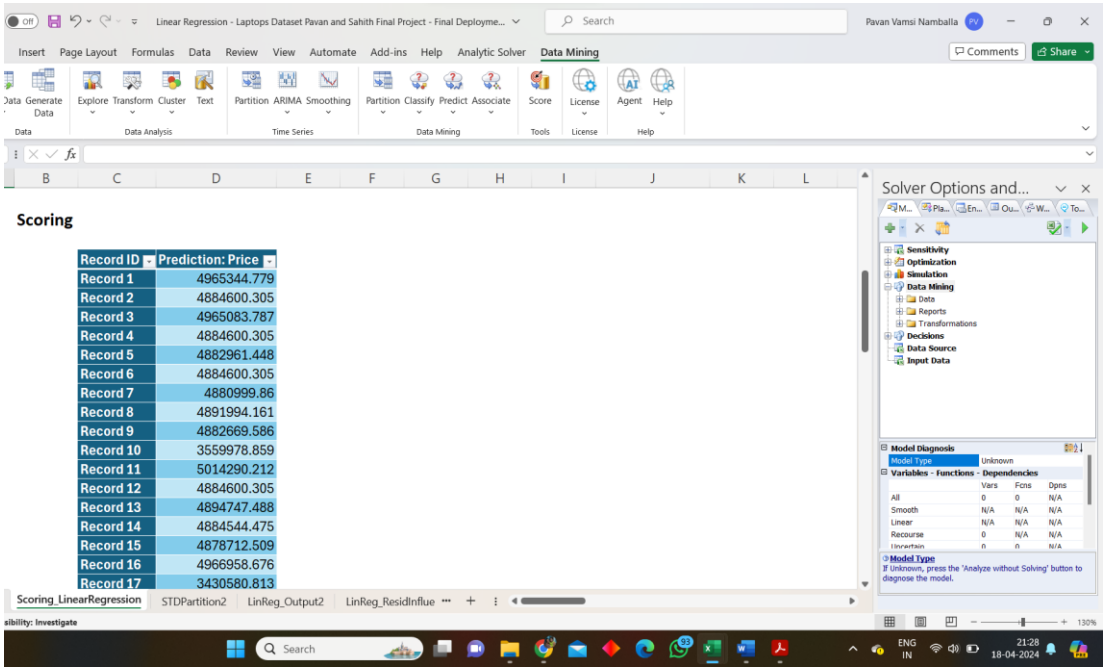
On clicking the **Score** button, a window with the name **Scoring** has opened. The **Model variables** and **variables in the new data** should be **matched by name**. Click on **Next**.



Click on **Finish**.



We now have a sheet called **Scoring_LinearRegression** where the predictions for the new data are displayed.



Scoring

Record ID	Prediction: Price
Record 1	4965344.779
Record 2	4884600.305
Record 3	4965083.787
Record 4	4884600.305
Record 5	4882961.448
Record 6	4884600.305
Record 7	4880999.86
Record 8	4891994.161
Record 9	4882669.586
Record 10	3559978.859
Record 11	5014290.212
Record 12	4884600.305
Record 13	4894747.488
Record 14	4884544.475
Record 15	4878712.509
Record 16	4966958.676
Record 17	3430580.813

Decision Trees

The Decision Tree is one of those algorithms which can be applied both for Classification as well as Regression. In the algorithm for Regression Trees, the fields in the training data are treated as nodes. A top-down tree is built with those nodes. Whenever the training data is fed into the machine learning system, it first looks for the field that can predict the output alone. Upon finding the perfect one, that node gets assigned as the Root Node. The records are classified into multiple subsets based on the root node. For each subset, the algorithm looks for that field which can predict the output alone. The prediction is in the form of an **average of the training examples in the terminal node**. The same process runs iteratively at each child node until we get a proper tree using which a proper set of rules are framed. These framed rules can be used in predicting the output for any new record from the validation dataset. Upon running the decision tree rules on the validation dataset, accuracy is measured. Two or more models are created with the same series of steps. The model with the highest validation accuracy R^2 and the lowest RMSE (Root Mean Squared Error) can be considered the best model.

Important Points: -

$$Number\ of\ Rules = Number\ of\ Terminals$$

$$Split\ Value = Midpoint\ of\ two\ consecutive\ Values$$

$$Branches\ represent\ the\ number\ of\ training\ examples$$

Brief explanation of the Algorithm: -

1. Pick One Predictor Variable, X_i.
2. Pick a Split Value S_i that can split X_i into two partitions, which need not be necessarily of equal size.
3. Split Value is nothing but the midpoint of the consecutive values. Split values are arranged based on the measure of homogeneity.
4. Homogeneity is the measure of how pure the class is. The node which alone can predict the outcome can be considered the “purest”. The node being purest indicates that all records in that node predict the same outcome.
5. Algorithm tries different values for X_i and S_i to maximize purity in each split.
6. After the successful split based on the Maximum purity, the same series of steps are followed for the next splits.

The rules learnt in the training data using steps 1 to 6 are now applied on the validation data. There might be some scenarios in the Validation Data which may not be available in the Training Data, which may lead to errors. If at all the Validation Data shows errors, the tree must be cut. The process is called Pruning. Pruning is done to avoid Overfitting.

Cost Complexity is the primary criterion for pruning. The tree with the lowest Cost Complexity is identified as the best one.

$$Cost\ Complexity = Proportion\ of\ Wrongly\ predicted\ Records + Penalty\ Factor\ attached\ to\ tree\ size(Set\ by\ the\ User)$$

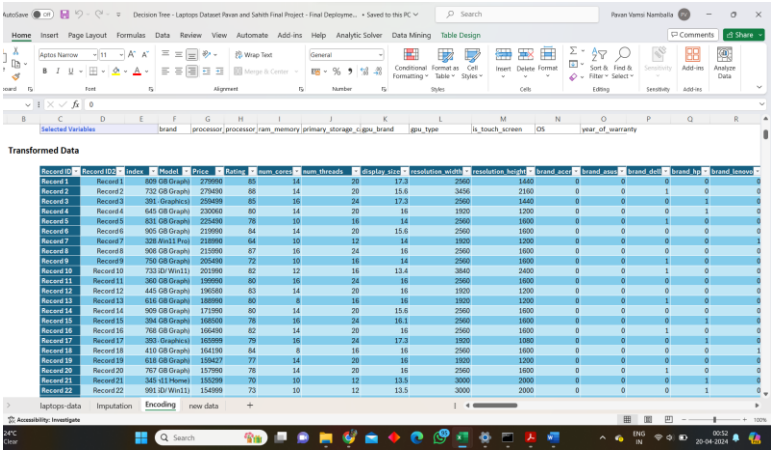
Pruning Results in two more trees:

1. **Minimum Error Tree:** Has lowest error rate on Validation Data.
2. **Best Pruned Tree:** Tree with lesser number of nodes. Error rate subject to constraint. Error must be less than certain extent.

It is the time the above algorithm can be applied on the laptops dataset to make predictions.

1. Clean Up the Data

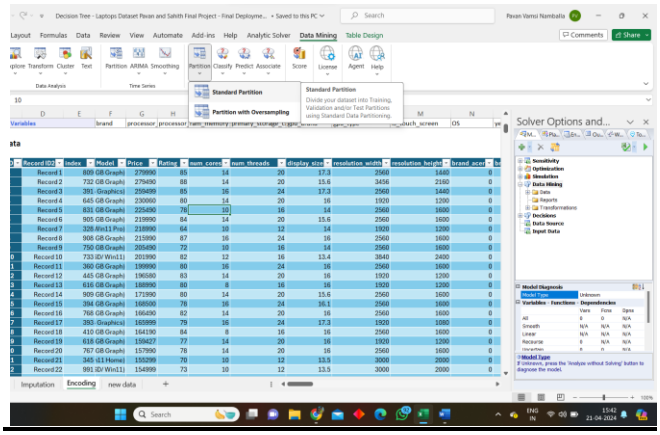
Using the same steps which were used in applying the **Linear Regression** algorithm, the data cleanup is done here. Upon doing so, we have the **Imputation** and the **Encoding** sheets created in our workbook **Decision Tree - Laptops Dataset Pavan and Sahith Final Project - Final Deployment**.



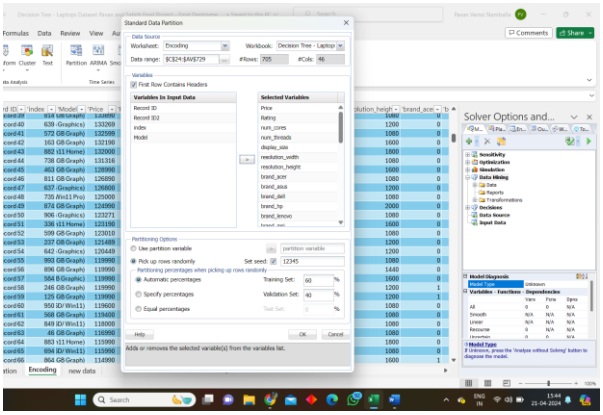
2. Time for the Partition

In this step, we divide the data into training and validation sets.
Click anywhere on the **encoding** sheet and go to the **Data Mining** tab.
Click on the **Partition** dropdown which is located left to the **Classify** dropdown.

Now click on **Standard Partition**.



The variables **index** and **model** which has no significance in determining the output are ignored along with the dynamically assigned variables **Record ID** and **Record ID2**.



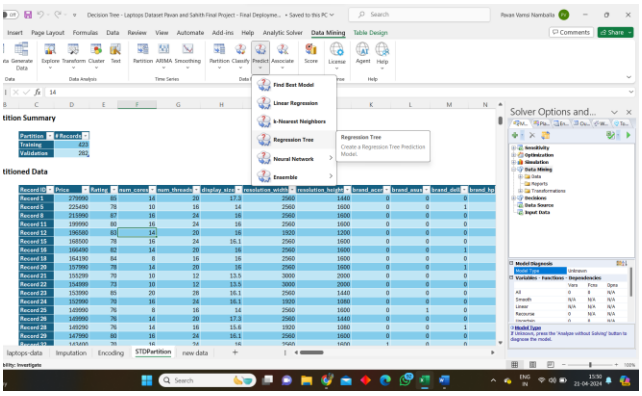
The ratio of training to validation data is **automatically set** to 60:40. To set manually, select the radio button **Specify Percentages**.

We have a new sheet created with the name **STDPartition**. It is on this sheet, where the algorithm is applied on.

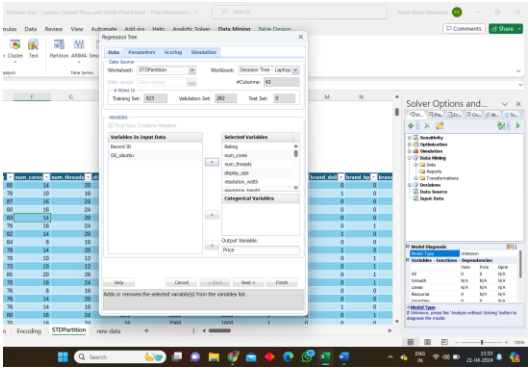
3. Time to Apply the Algorithm and evaluate the model!

Decision Tree is an algorithm that can be applied both for classification as well as regression, based on the nature of the output.

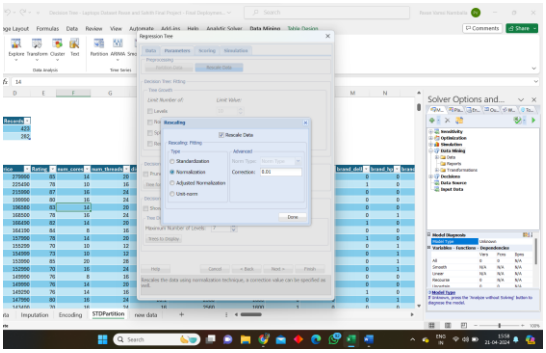
In our case, the output takes continuous numeric values. Hence, we select the one that is present under the **Predict** dropdown in the **Data Mining** tab.



The variable **Price** is sent to the **Output Variable** while the remaining ones are sent to the **Selected Variables**. Dynamically assigned variable **Record ID** is ignored by us while the variable **OS_ubuntu** is ignored by the system as it contains unary values. Click on **Next**.

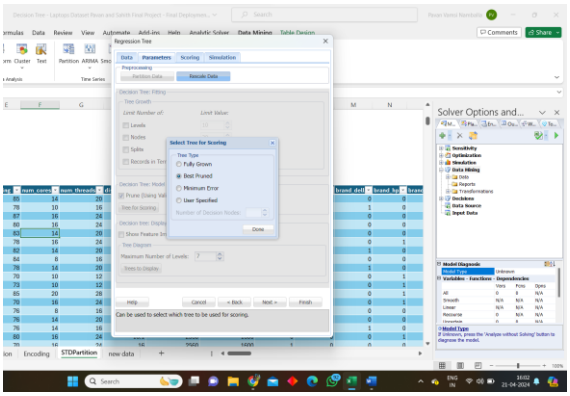


Under the **Regression Tree** window, go to **Parameters** tab, where the **data** is first **Rescale**d by the process of **Normalization**. Click on **Done**.



Upon rescaling, under the **Decision Tree: Model**, select the radio button **Prune** and click on the button **Trees for Scoring**. Select the checkbox **Best Pruned** and click on **Done**.

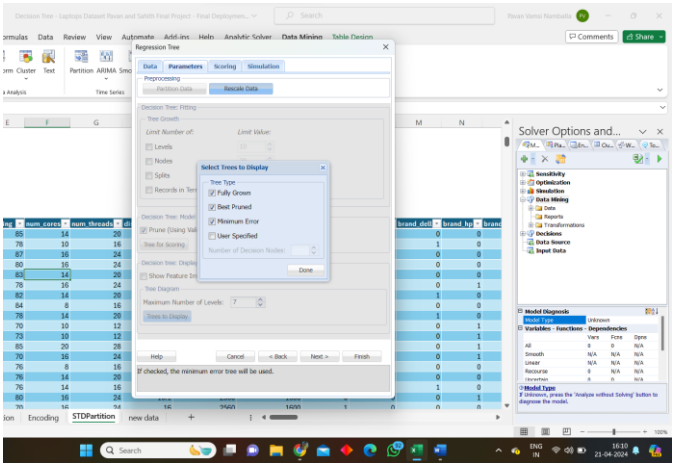
Best Pruned tree is used to evaluate the metrics of the model created and store it in a new table called the **Model Table**. Metrics of the model will be noted down in that table.



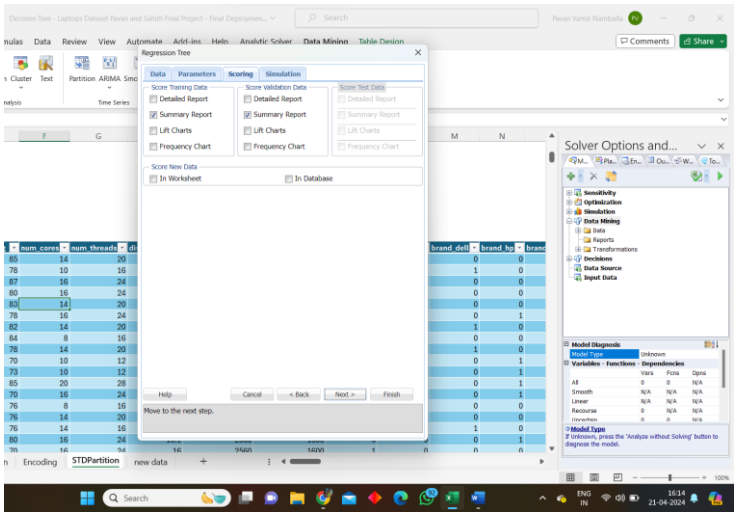
XLMiner can display three types of trees: **Best Pruned, Minimum Error, Fully Grown**.

1. **Fully Grown Tree: -** Tree Generated from all the examples in the Training Dataset.
2. **Best Pruned Tree: -** Tree with minimum number of nodes, subject to constraint set by the user. Pruned using Validation Data.
3. **Minimum Error Tree: -** Tree that has the least classification error rate when tested on the validation dataset.

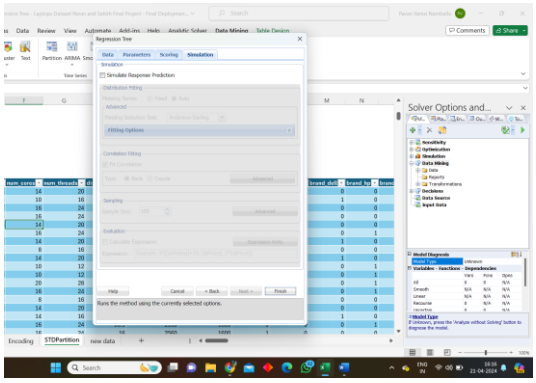
Click on **Trees to Display** button. Select all the three trees and click on **Done**.



Clicking on the **Next** button takes us to the **Scoring** tab. In the Scoring tab, **Summary Report** should be checked for both the **Training** as well as the **Validation Data**.

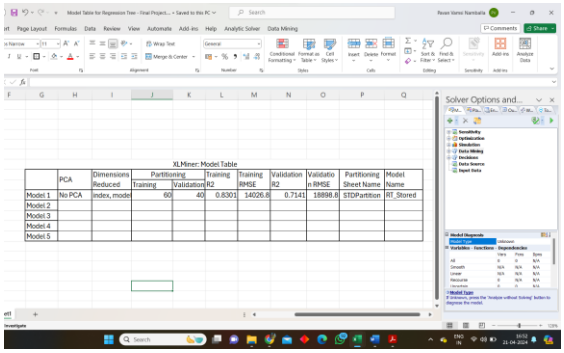


Clicking on the **Next** button takes us to the **Simulation** tab, where we need to click on **Finish**.



We have seven new sheets created with the names **RT_output, RT_FullTree, RT_BestTree, RT_MinErrorTree, RT_TrainingScore, RT_ValidationScore** and **RT_Stored**. **RT_Stored** is our model's name.

From the sheets **RT_TrainingScore** and **RT_ValidationScore**, we note down the values for **Training R2, Training RMSE, Validation R2** and **Validation RMSE** in the **Model Table**.



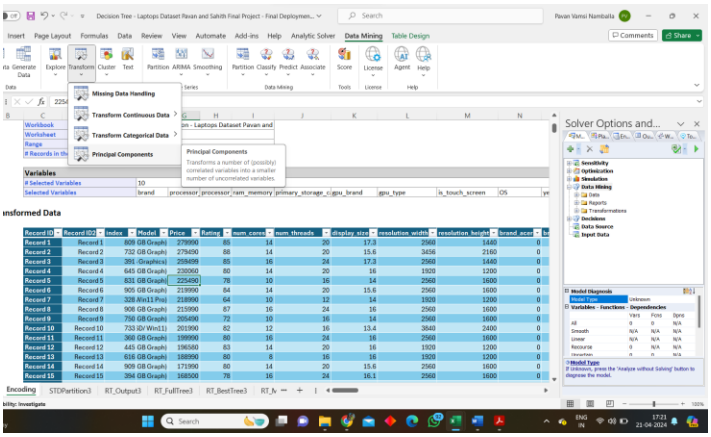
4. Create more Models!

It is time we created more models with different partition ratios for training and validation data. We create three models in such a way and note down the metrics in the same model table.

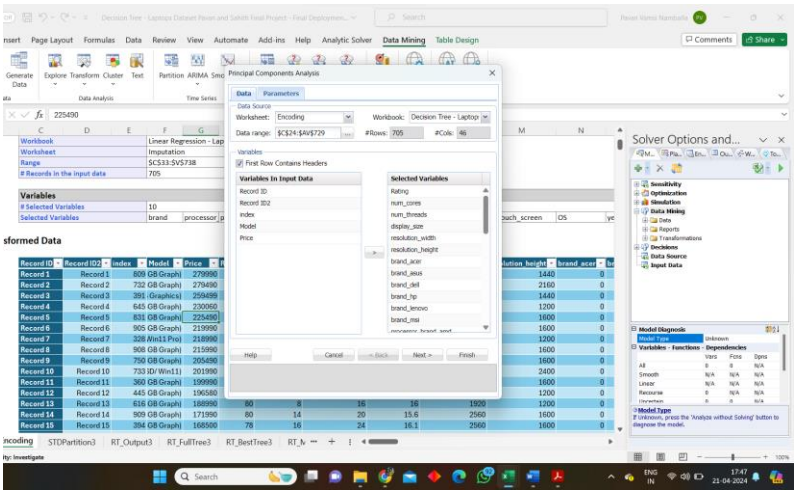
For the fifth model, we perform the Principal Component Analysis before applying the algorithm. PCA is done to know the top components that contribute to changes in the dataset.

Click anywhere on the **encoding** sheet and click on the **Data Mining** tab.

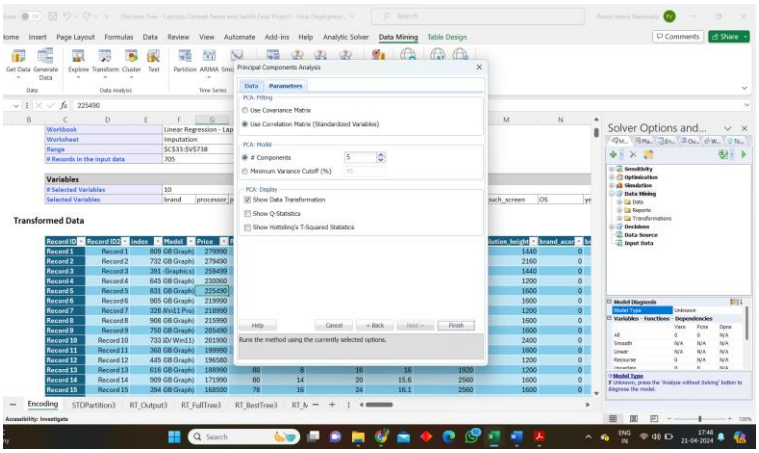
Go to **Transform -> Principal Components**



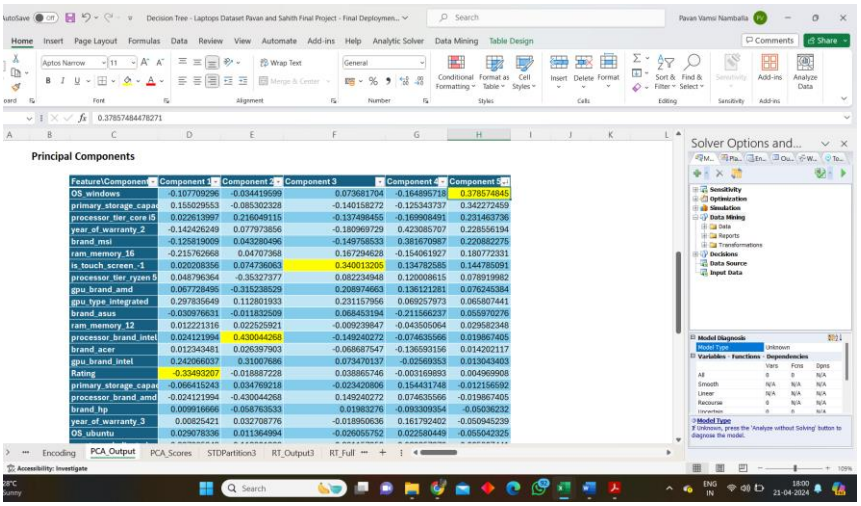
In the PCA, we ignore the output variable and include all the input variables.



We selected the **top 5 components** from the correlation matrix and clicked on **Finish** button.



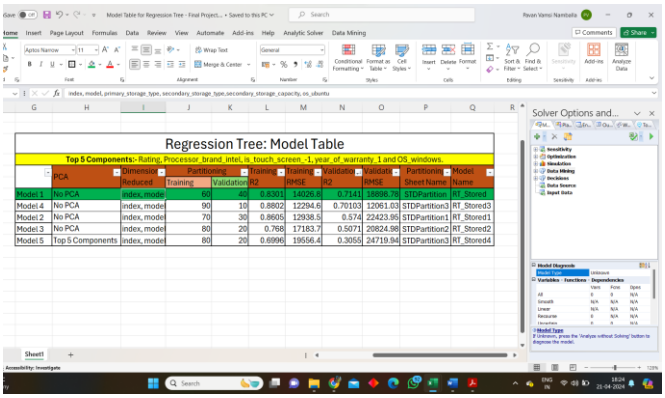
We have a sheet called **PCA_Output** where we have top 5 components displayed. They are: - **Rating**, **Processor_brand_intel**, **is_touch_screen_-1, year_of_warranty_1** and **OS_windows**.



We give these fields for data partition and apply the algorithm whose metrics are also noted down in the same model table.

5. Find the best model!

We give these fields for data partition and apply the algorithm whose metrics are also noted down in the same model table. A good Model should have a validation R^2 value of more than 70 percent and a low RMSE value. Just like in the case of Linear Regression, PCA doesn't work here as well.



Regression Tree: Model Table

Top 5 Components:- Rating, Processor_brand_intel, is_touch_screen_1, year_of_warranty_1 and OS_windows.

	PCA	Dimensio	Partitioning		Training	Training	Validatio	Validatio	Partitionin	Model
		Reduced	Training	Validation	R2	RMSE	R2	RMSE	Sheet Name	Name
Model 1	No PCA	index, model	60	40	0.8301	14026.5	0.7141	18886.78	STDPartition	RT_Stored
Model 4	No PCA	index, model	90	10	0.8802	12294.6	0.70103	12061.03	STDPartition3	RT_Stored3
Model 2	No PCA	index, model	70	30	0.8605	12938.5	0.574	22423.95	STDPartition1	RT_Stored1
Model 3	No PCA	index, model	80	20	0.788	17183.7	0.5071	20824.98	STDPartition2	RT_Stored2
Model 5	Top 5 Components	index, model	80	20	0.6996	19556.4	0.3055	24719.94	STDPartition3	RT_Stored4

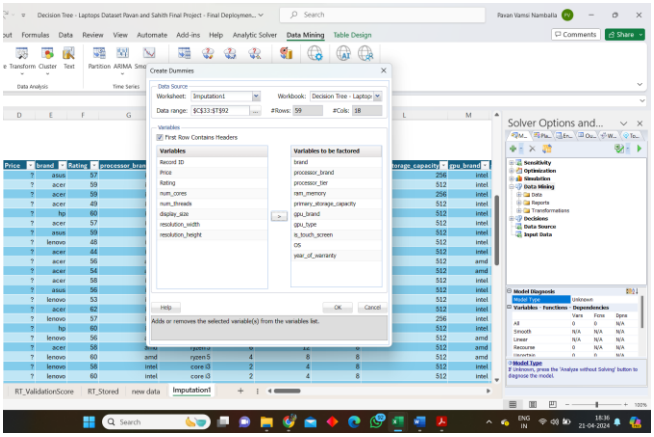
We can see that Models 1 and 4 follow that criteria. But the model 1, having the highest value for validation R^2 is considered the best model.

6. Apply the best model!

Scoring involves deploying the best model to make predictions for the records present in the test data present in the sheet called **new data**.

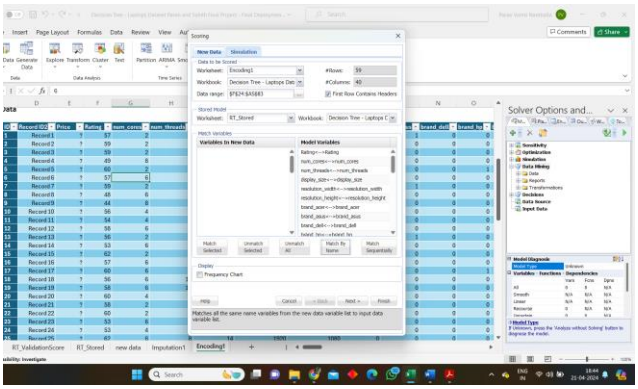
But, before doing so, we need to ensure that all the data is numerical.

We perform the **missing data handling**, followed by the **dummy creation** for the discrete data.



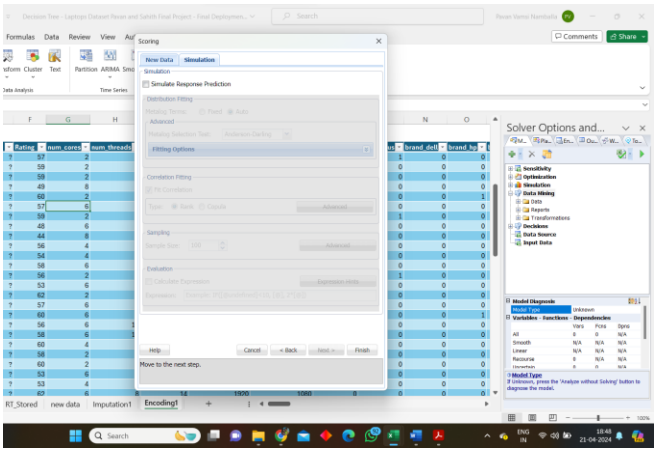
We have a new sheet called **imputation1**, created on performing the missing data handling. On this sheet, we create dummies, which can be seen in a new sheet called **encoding1**.

Click anywhere on the **encoding1** tab, and click **score** under the **Data Mining** tab.

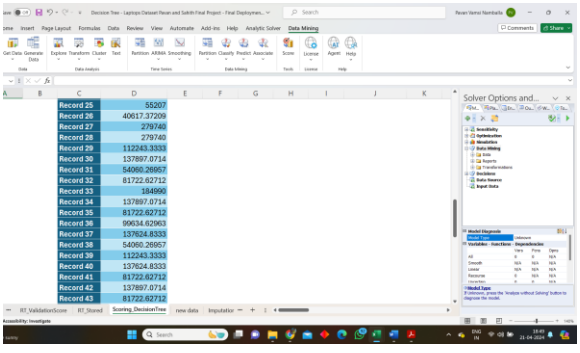


A new window appears with the name **Scoring**.
Under the **Stored Model**, there is a dropdown called the **worksheet** where the name of the appropriate model can be selected. We have already determined that **RT_Stored** is the best model. So, **RT_Stored** is applied.

Variables in the new data and the **Model Variables** should be **matched by name**. Click on **Next**.



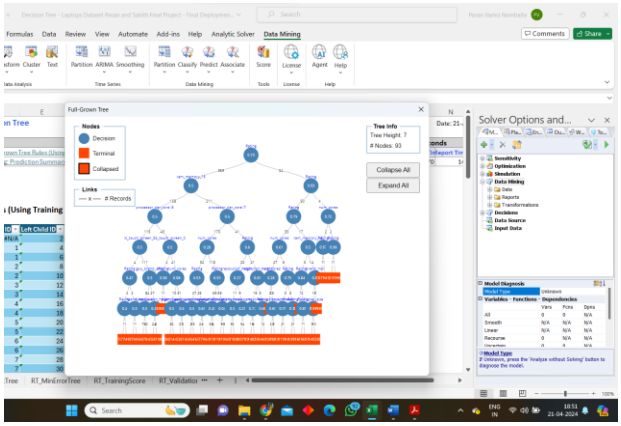
Under the **Simulation** tab, click on **Finish** button.



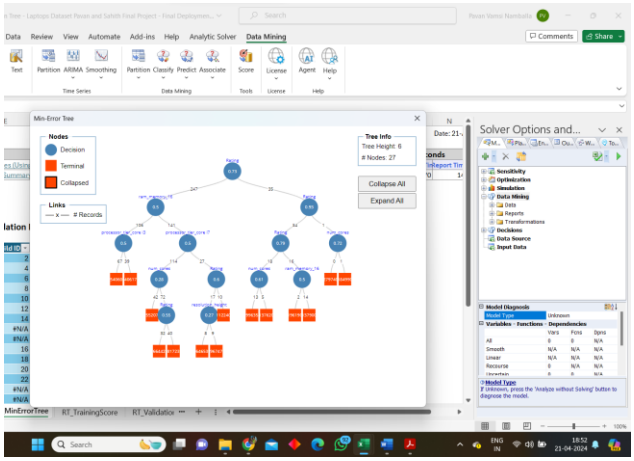
We have a new sheet called **Scoring_DecisionTree**, where we have the predictions available for the records in the **new data**.

Different Trees for the best model – RT_Stored: -

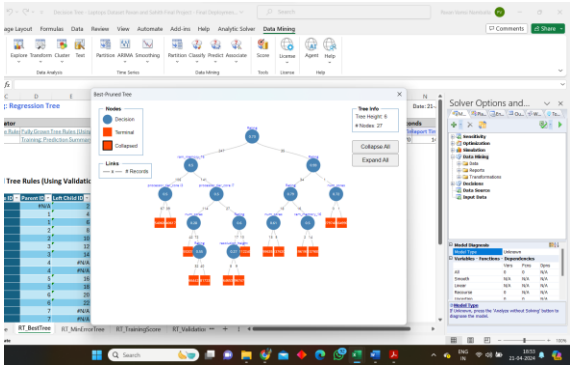
1. Fully Grown Tree



2. Minimum Error Tree



3. Best Pruned Tree



Comparing the three trees, the fully grown tree uses a larger number of variables while the remaining ones use only 6 variables: - **Rating, ram_memory_16, processor_tier_corei3, processor_tier_corei7, num_cores** and the **resolution_height**. This sort of analysis makes the **Regression Tree** algorithm suitable for the **Principal Component Analysis** as well.

Comparing the two Algorithms

We can clearly see that for the above dataset, the **Linear Regression** algorithm yields four models with a validation R^2 value of more than 70%, while the **Regression Tree** yields two models with a validation R^2 value just crossing 70%. It is also visible that the PCA for the top five components doesn't work for the above dataset in case of either of the algorithms. Based on the available evidence, we can conclude that the **Linear Regression** algorithm is the best for the above dataset.