

CSCU9M5 Practical 2

Model Performance

Part 1 - Metrics for Numeric Predictions (Regression)

You will need these files to perform the first part of this exercise:

[MotorPremiumsPredictions.csv](#)

[ExcelHistogram.xlsx](#)

In this exercise, we'll revisit the motor insurance claims and premiums data we looked at before. This time though, we've only been provided with a file containing a set of real values and the predictions made by a model for the same policies. This exercise will look at a few ways we can compare the two.

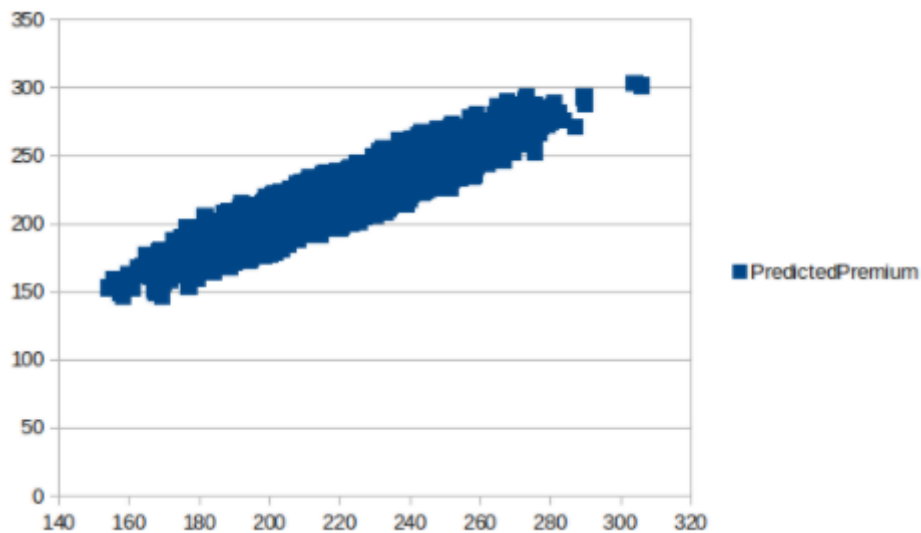
We'll be doing this in a spreadsheet. This will help you to see exactly where different figures are coming from, rather than just being magic numbers that Orange or Python are producing for you. You can use the spreadsheet of your choice: Excel, OpenOffice or LibreOffice will all do the job for you. Later in the course we'll be generating many of these measures and plots using the built-in tools.

In the following text, when you're asked to type in values, they are in quotes, "like this". You should omit the quotes and just type what's between them, followed by the enter key.

Open MotorPremiumsPredictions.csv in your spreadsheet and take a look. You can see there are two columns: Premium (the actual cost of the policy) and the predicted cost from a model. In general the two values look reasonably close.

1. Select columns A and B (as shown below), and insert a scatter plot. You can see how to do this in Excel [here](#); and in LibreOffice [here](#). The plot should look something like the one you can see below. In general, the predictions seem to be roughly in line with the actual premiums. Let's try a few ways to quantify that.

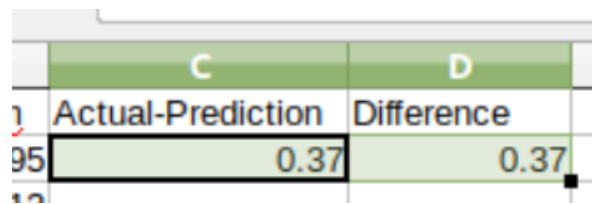
	A	B	C
1	Premium	PredictedPremium	
2	195.32	194.95	
3	225.09	222.13	
4	201.08	221.98	
5	211.76	211.24	
6	251.49	271.62	
7	229.46	217.95	
8	255.25	248.66	
9	229.13	223.73	
10	214.8	222.45	
11	187.77	187.46	
12	235.86	246.06	



2. In cell C1, type “Actual-Prediction”, then in C2, type “=A2-B2”. In cell D1, type “Difference”, then D2 “=abs(c2)”. When you press enter on these, you’ll see that both cells have the value 0.37: the difference between the actual premium of 195.32 and predicted premium of 194.95.

	A	B	C	D
1	Premium	PredictedPremium	Actual-Prediction	Difference
2	195.32	194.95	0.37	0.37
3	225.09	222.13	2.96	2.96
4	201.08	221.98	-20.9	20.9
5	211.76	211.24	0.52	0.52
6	251.49	271.62	-20.13	20.13
7	229.46	217.95	11.51	11.51
8	255.25	248.66	6.59	6.59
9	229.13	223.73	5.4	5.4
10	214.8	222.45	-7.65	7.65
11	187.77	187.46	0.31	0.31
12	235.86	246.06	-10.2	10.2
13	198.68	190.43	8.25	8.25

3. Drag a box around C2 and D2, like this:



	C	D
1	Actual-Prediction	Difference
2	0.37	0.37

Then double click on the “handle” at the bottom right of this box (the black square in the illustration). This will copy the formulas into all the cells below. The first few lines of the spreadsheet should now look like the illustration on the right. You can see that some of the predictions are too high, some too low, and in general they are up to about 10 out from where they should be.

4. Unfortunately there isn’t an easy way to generate a histogram in any of the common spreadsheet packages. We’ve provided a separate spreadsheet that will approximate a histogram for you. Select the values in cells D2 to D4388, and copy them to your clipboard. Then open ExcelHistogram.xlsx, select cell A3, and right-click, then “paste special” and either “values” (Excel) or “Numbers” (Open/LibreOffice). This will paste the differences into this new spreadsheet, and generate a plot that should look like the one on the right. You can see that the difference for the majority of the predictions is less than 5. You can try doing the same thing with the “Actual-Prediction” values and you’ll get a rather satisfying bell curve. These plots will give you a good idea of how often the prediction errors exceed a certain amount.

5. In cell F2, type “MAE:”. In cell G2, type “=AVERAGE(D2:D4388)”. This is the Mean Absolute Error, and should come to just over 6.26.

6. What happens if we add a similar formula in cell F3 to calculate the average of the values in column C?

7. Fill column E with the squares of the values in column C. You can do this with the formula “=C2*C2”, and drag out a box to fill the column as before. Now calculate the average of column E in cell F4. This is the Mean Square Error. Does this have much meaning from a human perspective?

8. In cell G5, enter the formula “=CORREL(A:A,B:B)”. This is the correlation between predicted and true values, and typically a value over about 0.7 indicates a strong positive correlation. Is that the case here? So, overall, would you say that the predictive model generating these values is good?

Part 2 - Metrics for Categorical Predictions (Classification)

You will need this file to perform this part of this exercise.

[MotorClaimsPredictions.csv](#)

Open MotorClaimsPredictions.csv in your spreadsheet and take a look. You can see there are two columns: Claim (indicating if a claim was made against the policy) and the prediction from

a model. As before, it looks like the model has got the result correct most of the time. In fact, in the first 20 rows, only two of them are wrong (a “no” instead of a “yes”), so the accuracy (the percentage of predictions that were correct) is 2 out of 20, or 10%.

1. Let’s calculate the accuracy for the whole data set. In cell C1, type “Correct”, and in C2, “=A2=B2”. Drag a box to fill in the C column as you did before. Now, the values in C show TRUE if the prediction was correct, and FALSE if it was wrong.

2. In cell G2, type “Count correct:”, and in H2, “=COUNTIF(C:C,TRUE)”. This will display a count of the total number of correct predictions, by counting how often “TRUE” appears

3. In cell G3, type “Count total:”, and in H3, “4387”. We know there are 4387 predictions in this set so we can just type the number.

4. In cell G4, type “Accuracy:”, and in H4, “=H2/H3”. We get an accuracy figure of 0.89 for the data set. Is this a good model? What if we care more about predicting accurately when a claim will happen rather than when one won’t happen?

5. Starting in cell G6, enter the formulae as shown below. What’s going on here is that we’re counting all rows where the value in column A and column B take particular values. So the middle cell in this table is counting the times the actual value and the predicted value were both “Yes”. Once you’ve entered these, you can see how often the model is getting the “Yes” and “No” predictions right. Which class is it better at predicting?

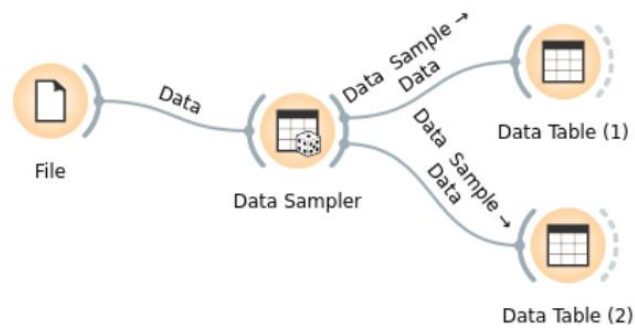
	Predicted: No	Predicted: Yes
Actual: No	=COUNTIFS(A:A,"No",B:B,"No")	=COUNTIFS(A:A,"No",B:B,"Yes")
Actual: Yes	=COUNTIFS(A:A,"Yes",B:B,"No")	=COUNTIFS(A:A,"Yes",B:B,"Yes")

6. If you add the “predicted no/actual no” and “predicted yes/actual yes” figures from the above table, it should come to the same value as the “Count correct” figure from earlier. So, this table, the Confusion Matrix for our model, also contains that more general measure.

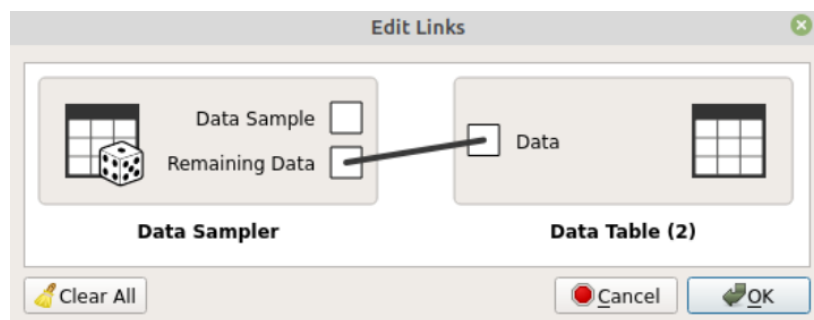
In this exercise, we’ve tried generating some basic analysis of the quality of a model’s predictions. In general, this will be done for you by Orange or sk-learn, but inside these tools, the same basic calculations are happening.

Train/Test Split

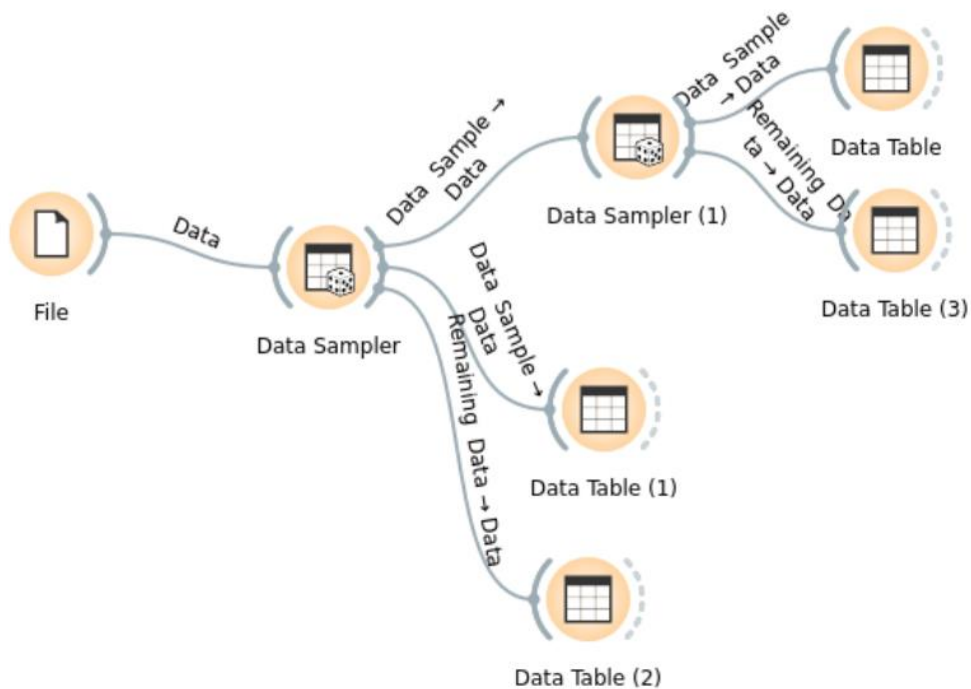
We’ll now take a look at how to split the data into training, test and validation sets in Orange. The clearest way is using the Data Sampler widget that we met in the first session. To begin with, create a new Orange workflow, add a File widget, and open the MotorPremiums.csv file. Double click on the File widget, and for Premium, set the Role to “target”. The line for Premium should go bold: this is telling Orange that when we get to making models it’s the “Premium” we’ll be predicting. Close the File widget, and connect a Data Sampler to it. Open the data sampler and tell it you want to have a fixed proportion of data of 80%. Click on “Sample Data” then close the Data Sampler. To the Data Sampler, connect two Data Table widgets like this:



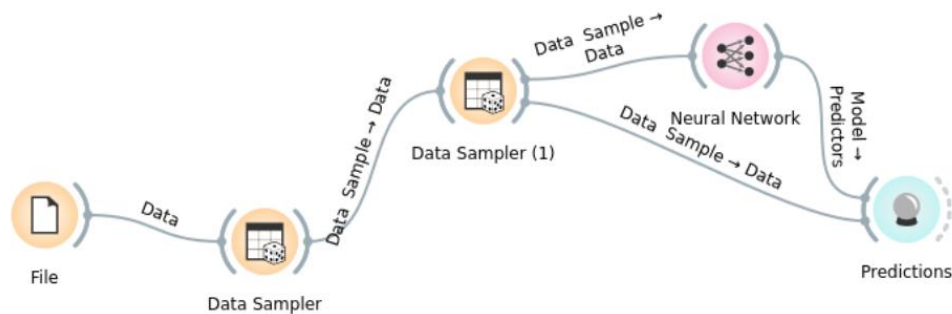
Then double click on the line connecting to the second Data Table and change the connections to look like this:



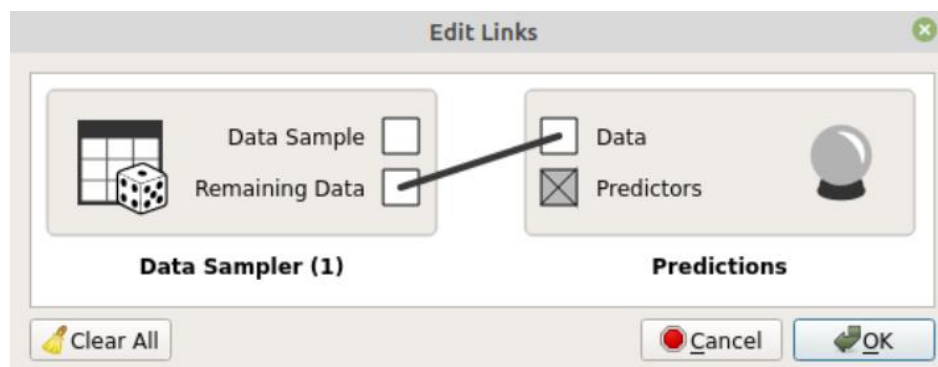
What we have done is split the data, at random, into two sets. The top one should have 3510 data points, and is our training set. The bottom one, with 877 data points, is our test set. We can connect another Data Sampler, to split the training data into training and validation sets, like this:



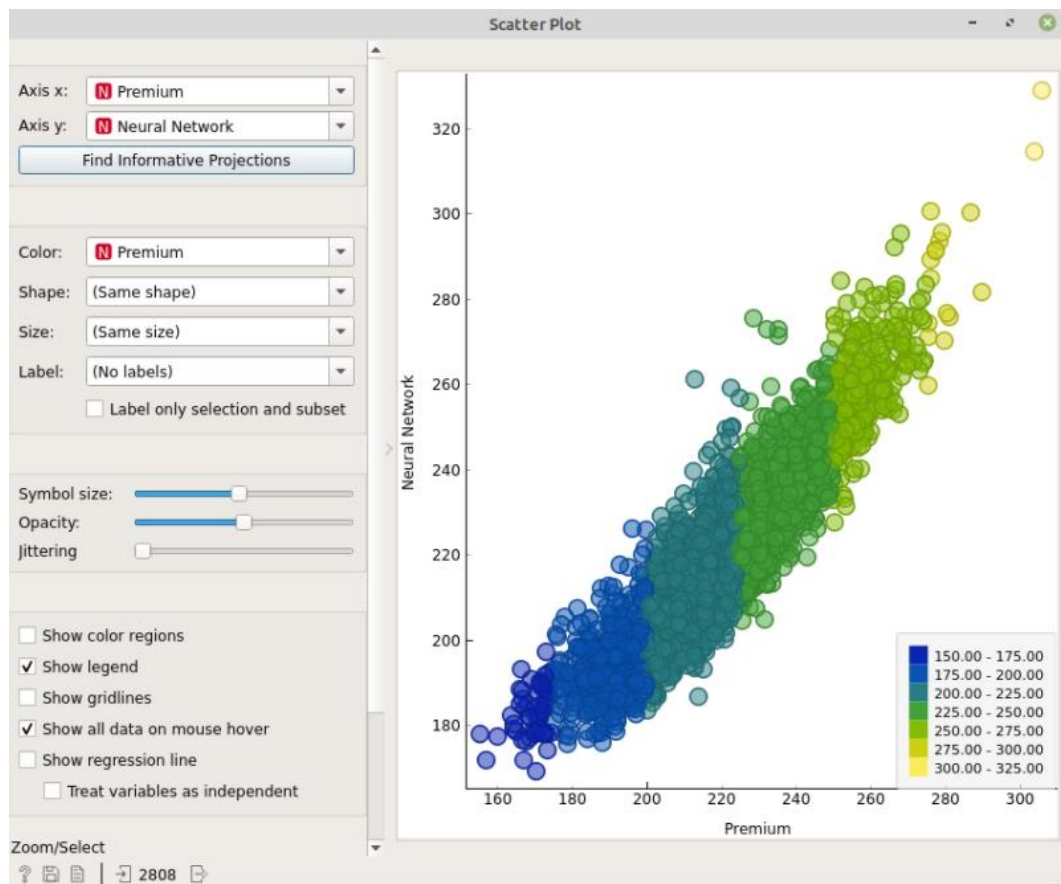
The data tables above are really just to let you see the data after it's been split and aren't needed - so you can delete them again now. Let's say we want to actually train a model on this data... we've not really met any specific models yet so can just regard it as a "magic box" that makes predictions at the moment. Add a Neural Network to the second Data Sampler. It will show a progress bar as it trains on the training data. Once that's done, you've just made your first predictive model! Connect a Predictions widget to the Neural Network, and draw a line connecting the Data Sampler to Predictions like this:



The Predictions widget takes a trained model and applies it to some new data. Make sure that Predictions is receiving the "validation" data (i.e. the remaining 20% that wasn't used to train the model):

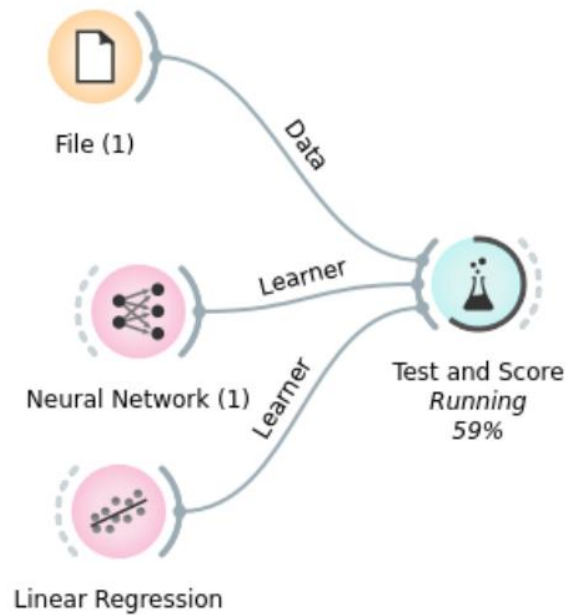


You can now take a look at the predictions. If you connect a Scatter Plot to the right side of Predictions, and set axis X and Y to Premium and Neural Network, you can see how well the neural network's predictions related to the true premiums:



Similarly you can use a Correlations widget to see how the predictions from the neural network correlate with the true premium values; or you can use a Data Table to directly see what the predictions were.

This is all good, but in practice you probably want to use cross fold validation. Orange provides a “Test and Score” widget, which does a lot of the hard work for you. If you supply it with an untrained model and a data set, it will split the data for you and compute the most commonly used metrics. You can tell it to do a single split of the data as we did above, or you can tell it to do cross fold validation. Here is a workflow using Test and Score:



Here we've actually supplied two untrained models (a Neural Network and a Tree) so we can compare them. Open the Test and Score widget:

Test and Score

Sampling

- ☒ Cross validation
 - Number of folds: 5
 - ☒ Stratified
- ☐ Cross validation by feature
- ☐ Random sampling
 - Repeat train/test: 10
 - Training set size: 95 %
 - ☒ Stratified
- ☐ Leave one out
- ☐ Test on train data
- ☐ Test on test data

Model Comparison

Mean square error

☐ Negligible difference: 0.1

Evaluation Results

Model	MSE	RMSE	MAE	R2
Neural Network	56.344	7.506	5.609	0.886
Linear Regression	19.995	4.472	3.565	0.959

Model Comparison by MSE

	Neural Net...	Linear Regr...
Neural Network		0.997
Linear Regression	0.003	

Table shows probabilities that the score for the model in the row is higher than that of the model in the column. Small numbers show the probability that the difference is negligible.

Here we've used 5-fold cross validation. You can see that changing the number changes how long the models take to retrain. Likewise you can change to a train/test split using "Random sampling"; or you can connect another data source (either from a file or from a Data Sampler) and calculate the model performance on that using "Test on test data".

On the right you can see the models compared in terms of the average scores over the 5 folds for several different measures. The widget has chosen measures appropriate for a regression problem (they are different for classification problems). In this example, linear regression seems to be a better model because its Mean Square Error is lower and its R-squares value is higher.

In practice, you'll be using Test and Score a lot as it is a very convenient way to compare the models.

You will learn more about neural networks and regression in later weeks of this module.