

BA Excel Add-In User's Manual (Version 0.0.32)

Contents

1	The BA Add-In Functions	3
1.1	Linear Regression Functions	3
1.1.1	LinearRegTrain	3
1.1.2	LinearRegPredict	6
1.2	Logistic Regression Functions	8
1.2.1	LogisticRegTrain	8
1.2.2	LogisticRegPredict	12
1.3	K-Nearest Neighbor Functions (Content-based)	15
1.3.1	knn_in	15
1.3.2	knn_in_rmse	17
1.3.3	knn_out	19
1.4	K-Nearest Neighbor Functions (User-based)	21
1.4.1	knn_in_movie	21
1.4.2	knn_in_movie_rmse	23
1.5	Other KNN Functions	24
1.5.1	knn_dist	24
1.5.2	knn_nearest	26
1.6	BA_RMSE	27
1.7	Classification Functions	29
1.7.1	confusionMatrix	29

1.7.2	ROC	31
1.7.3	AUC	34
1.7.4	ROC (with cost information)	34
1.8	Monte Carlo Simulation	35

1. The BA Add-In Functions

1.1. Linear Regression Functions

1.1.1. LinearRegTrain

The example illustrated below is available on the *add-in webpage* by downloading the *Linear and Logistic Regression file* and going to the *linearReg* sheet.

LinearRegTrain is an array function that allows you to estimate the coefficients $(\beta_0, \beta_1, \dots, \beta_K)$ of a regression equation of the form

$$Y = \beta_0 + \beta_1 \text{Var1} + \beta_2 \text{Var2} + \dots + \beta_K \text{VarK} + \epsilon,$$

using observed data on the independent variables (**Var1**, **Var2**, ..., **VarK**) and on the dependent variable **Y**. This function outputs the estimated coefficients, the standard errors, the associated p-values, and the R-squared of the regression.

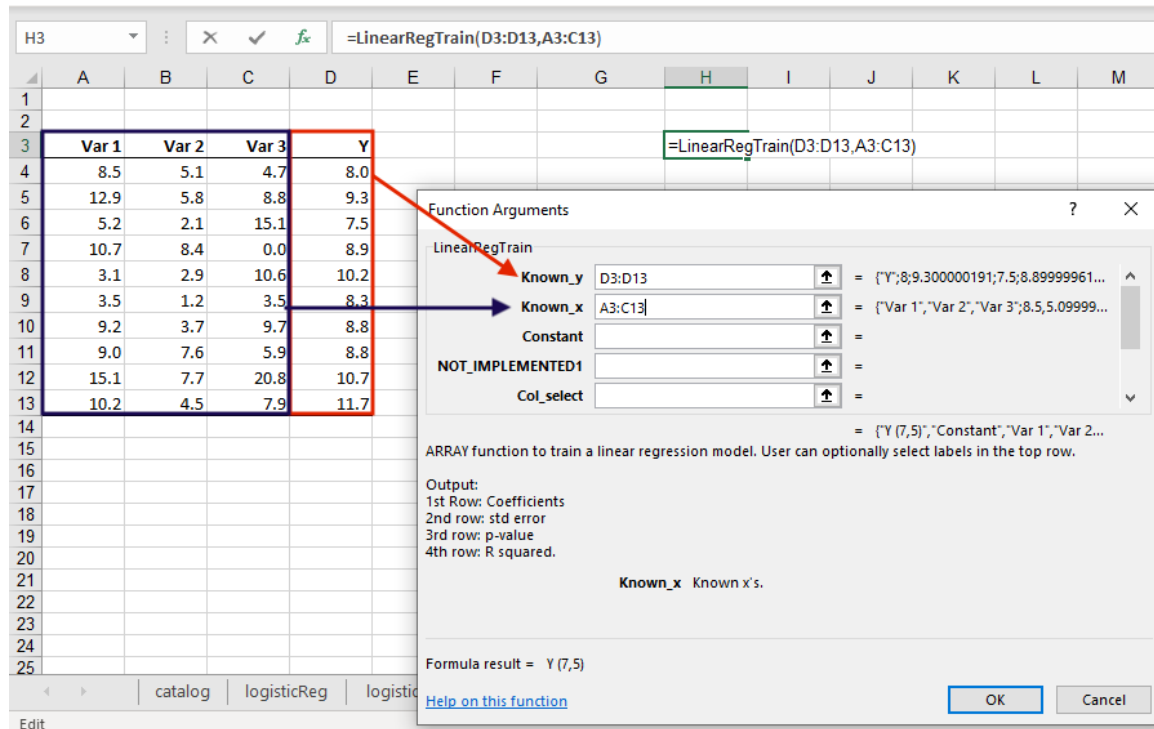
We illustrate how this function works with a dataset that contains 10 observations of 3 independent variables (**Var1**, **Var2**, **Var3**) and the dependent variable **Y**.

To use the function:

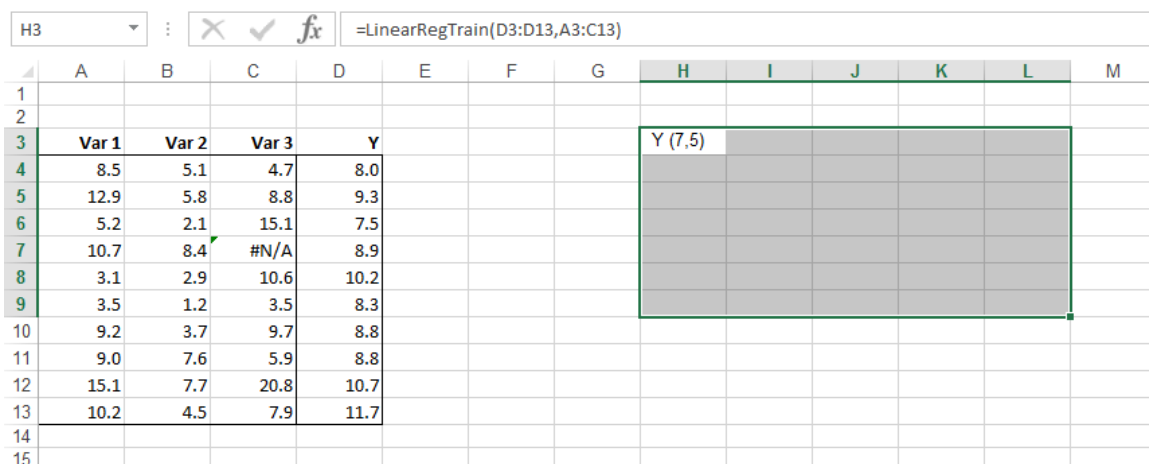
1. Select the cell where you want to display the output (H3 in the example) and call the function by typing `=LinearRegTrain(` and then clicking on the f_x symbol next to the formula bar. (Alternatively, you can call the function by clicking on the f_x symbol first, which will display a menu of all available functions, and then clicking on **LinearRegTrain**.)

H3											
	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3	Var 1	Var 2	Var 3	Y				=LinearRegTrain(
4	8.5	5.1	4.7	8.0							
5	12.9	5.8	8.8	9.3							
6	5.2	2.1	15.1	7.5							
7	10.7	8.4	0.0	8.9							
8	3.1	2.9	10.6	10.2							
9	3.5	1.2	3.5	8.3							
10	9.2	3.7	9.7	8.8							
11	9.0	7.6	5.9	8.8							
12	15.1	7.7	20.8	10.7							
13	10.2	4.5	7.9	11.7							
14											
15											
16											
17											

- When you call the function, the window **Function Arguments** will appear. Follow the instructions to populate the input fields, then click OK.



- The string `Y (7,5)` will appear in cell H3, where Y is the name of the variable in cell D3. This indicates that you should select an output array of 7 rows and 5 columns. Select the output array starting from cell H3 as the next figure shows.



- Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the output. *Remark: for recent versions of Excel, this step is not necessary.*

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3	Var 1	Var 2	Var 3	Y				Y (7,5)	Constant	Var 1	Var 2	Var 3
4	8.5	5.1	4.7	8.0				Coefficients	7.837367	0.120385	-0.00744	0.042174
5	12.9	5.8	8.8	9.3				Std error	1.292461	0.229497	0.348021	0.091981
6	5.2	2.1	15.1	7.5				p-value	0.09%	61.87%	98.36%	66.27%
7	10.7	8.4	0.0	8.9				R-sqr	0.197115			
8	3.1	2.9	10.6	10.2				Number valid obs	10			
9	3.5	1.2	3.5	8.3				Total obs	10			
10	9.2	3.7	9.7	8.8								
11	9.0	7.6	5.9	8.8								
12	15.1	7.7	20.8	10.7								
13	10.2	4.5	7.9	11.7								
14												
15												

The second row of the output contains the estimates of the coefficients: the column (**Constant**) refers to the coefficient β_0 , while the columns (**Var1,Var2,Var3**) refer to the coefficients ($\beta_1, \beta_2, \beta_3$) respectively. In the following rows we have standard errors and p-values of the estimates as well as the R^2 of the regression. The last two rows contain the number of valid observations and the total number of observations respectively.

You can also format the output to display only three decimals and to display percentages.

Optional Input: Col_select. The optional input **Col_select** allows you to specify which variables you want to include in your regression. For example, the configuration below runs a logistic regression with only **Var1** and **Var3** as independent variables.

	A	B	C	D	E	F	G	H	I	J	K	L
1	1	0	1	column select (1 include, 0 exclude)								
2												
3	Var 1	Var 2	Var 3	Y				A1:C1	Constant	Var 1	Var 2	Var 3
4	8.5	5.1	4.7	8.0								
5	12.9	5.8	8.8	9.3								
6	5.2	2.1	15.1	7.5								
7	10.7	8.4	0.0	8.9								
8	3.1	2.9	10.6	10.2								
9	3.5	1.2	3.5	8.3								
10	9.2	3.7	9.7	8.8								
11	9.0	7.6	5.9	8.8								
12	15.1	7.7	20.8	10.7								
13	10.2	4.5	7.9	11.7								
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												

Function Arguments

LinearRegTrain

Known_y D3:D13 = {"Y";8.300000191;7.5;8.89999961...

Known_x A3:C13 = {"Var 1","Var 2","Var 3";8.5;5.09999...

Constant NOT_IMPLEMENTED1 =

Col_select A1:C1 = {1,0,1}

= {"Y (7,4)","Constant","Var 1","Var 3...

ARRAY function to train a linear regression model. User can optionally select labels in the top row.

Output:
1st Row: Coefficients
2nd row: std error
3rd row: p-value
4th row: R squared.

Col_select A vector of 0/1. If omitted, all columns in Known_x are considered for regression.If specified, and if the jth entry is 0 in this vector, the jth column in Known_x matrix is ignored.

Formula result = Y (7,4)

[Help on this function](#)

OK Cancel

1.1.2. LinearRegPredict

The example illustrated below is available on the *add-in webpage* by downloading the *Linear and Logistic Regression* file and going to the *linearReg* sheet.

LinearRegPredict uses the coefficients ($\beta_0^e, \beta_1^e, \dots, \beta_K^e$) that we estimate with **LinearRegTrain** and the data on the independent variables (**Var1**, **Var2**, ..., **VarK**) to compute a prediction for the dependent variable using the formula

$$Y^P = \beta_0^e + \beta_1^e \text{Var1} + \beta_2^e \text{Var2} + \dots + \beta_K^e \text{VarK}.$$

For example, using the coefficients that we estimated in the **LinearRegTrain** example we would predict **Y** using the following formula

$$Y^P = 7.854 + 0.114\text{Var1} + 0.023\text{Var2} + 0.034\text{Var3},$$

which, for the first observation, yields a prediction of

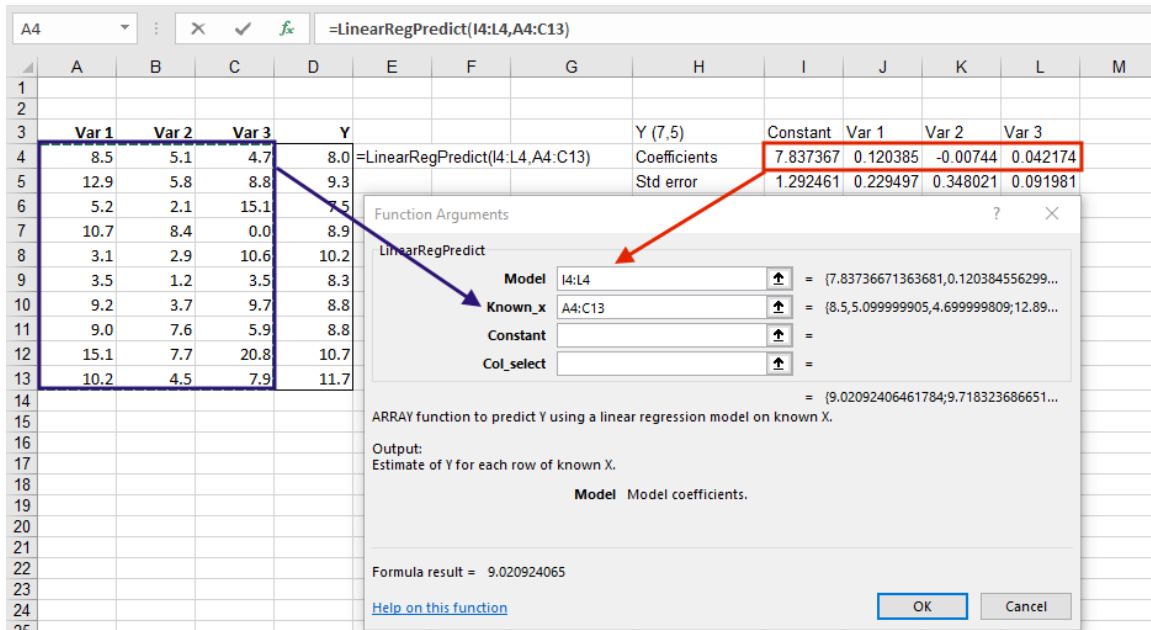
$$9.1001 = 7.854 + 0.114 * 8.5 + 0.023 * 5.1 + 0.034 * 4.7.$$

To use the function:

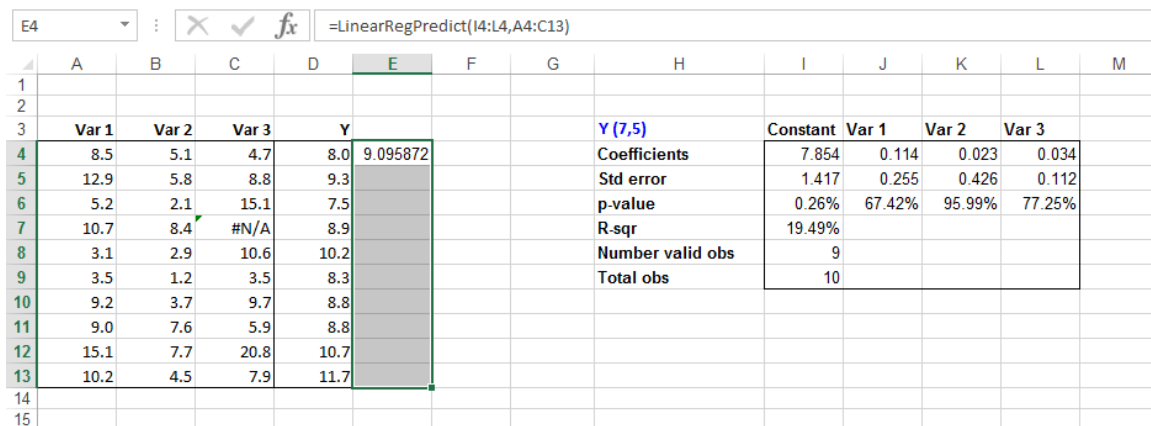
1. Select the first cell in which you want the output to be displayed; call the function by typing **=LinearRegPredict(** in the formula bar and then pressing the f_x symbol next to the formula bar.

SUM	:	X	✓	<i>f_x</i>	=LinearRegPredict(
1	A	B	C	D	E	F	G	H	I	J	K	L	M
2													
3	Var 1	Var 2	Var 3	Y				Y (7,5)	Constant	Var 1	Var 2	Var 3	
4	8.5	5.1	4.7	8.0	=LinearRegPredict(Coefficients	7.837367	0.120385	-0.00744	0.042174	
5	12.9	5.8	8.8	9.3				Std error	1.292461	0.229497	0.348021	0.091981	
6	5.2	2.1	15.1	7.5				p-value	0.09%	61.87%	98.36%	66.27%	
7	10.7	8.4	0.0	8.9				R-sqr	0.197115				
8	3.1	2.9	10.6	10.2				Number valid obs	10				
9	3.5	1.2	3.5	8.3				Total obs	10				
10	9.2	3.7	9.7	8.8									
11	9.0	7.6	5.9	8.8									
12	15.1	7.7	20.8	10.7									
13	10.2	4.5	7.9	11.7									
14													
15													
16													

2. Follow the instructions to populate the input fields, as demonstrated in the figure below, then click OK.



- Remember that this is an array function. Select the column in which you want the output to be displayed: this should be a column of the same length as the number of observations that will contain the predictions associated to each observation. In the figure below, we have 10 observations and we select cells E4:E13.



- Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the predictions. You can also add a title (for example "LinearRegPredict") to the output-column and format the output-cells to display only the first 3 decimals, as in the figure below. *Remark: for recent versions of Excel, this step is not necessary.*

E4													
	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3	Var 1	Var 2	Var 3	Y				Y (7,5)	Constant	Var 1	Var 2	Var 3	
4	8.5	5.1	4.7	8.0	9.020924			Coefficients	7.837367	0.120385	-0.00744	0.042174	
5	12.9	5.8	8.8	9.3	9.718324			Std error	1.292461	0.229497	0.348021	0.091981	
6	5.2	2.1	15.1	7.5	9.084577			p-value	0.09%	61.87%	98.36%	66.27%	
7	10.7	8.4	0.0	8.9	9.063009			R-sqr	0.197115				
8	3.1	2.9	10.6	10.2	8.636036			Number valid obs	10				
9	3.5	1.2	3.5	8.3	8.397397			Total obs	10				
10	9.2	3.7	9.7	8.8	9.326475								
11	9.0	7.6	5.9	8.8	9.113132								
12	15.1	7.7	20.8	10.7	10.47513								
13	10.2	4.5	7.9	11.7	9.364997								
14													
15													
16													

Optional Input: Col_select. See explanation at the end of Section 1.1.1.

1.2. Logistic Regression Functions

1.2.1. LogisticRegTrain

The example illustrated below is available on the add-in webpage by downloading the Linear and Logistic Regression file and going to the logisticReg sheet.

LogisticRegTrain is an array function that allows you to estimate the coefficients ($\beta_0, \beta_1, \dots, \beta_K$) of a logistic regression model using observed data on the independent variables (**Var1, Var2, ..., VarK**) and on the dependent variable **Y**. Note that the dependent variable in a logistic regression must be binary, i.e. it must take values 0 or 1. This function outputs the estimated coefficients, the standard errors, and the associated p-values. This function allows you to estimate the probability that the dependent variable is equal to 1 as

$$Pr(\text{dependent variable equals one}) = \frac{\exp(w)}{1 + \exp(w)},$$

with

$$w = \beta_0 + \beta_1 \text{Var1} + \beta_2 \text{Var2} + \dots + \beta_K \text{VarK}.$$

We illustrate how this function works with a dataset that contains 18 observations of 3 independent variables (**Var1, Var2, Var3**) and the dependent variable **Y**.

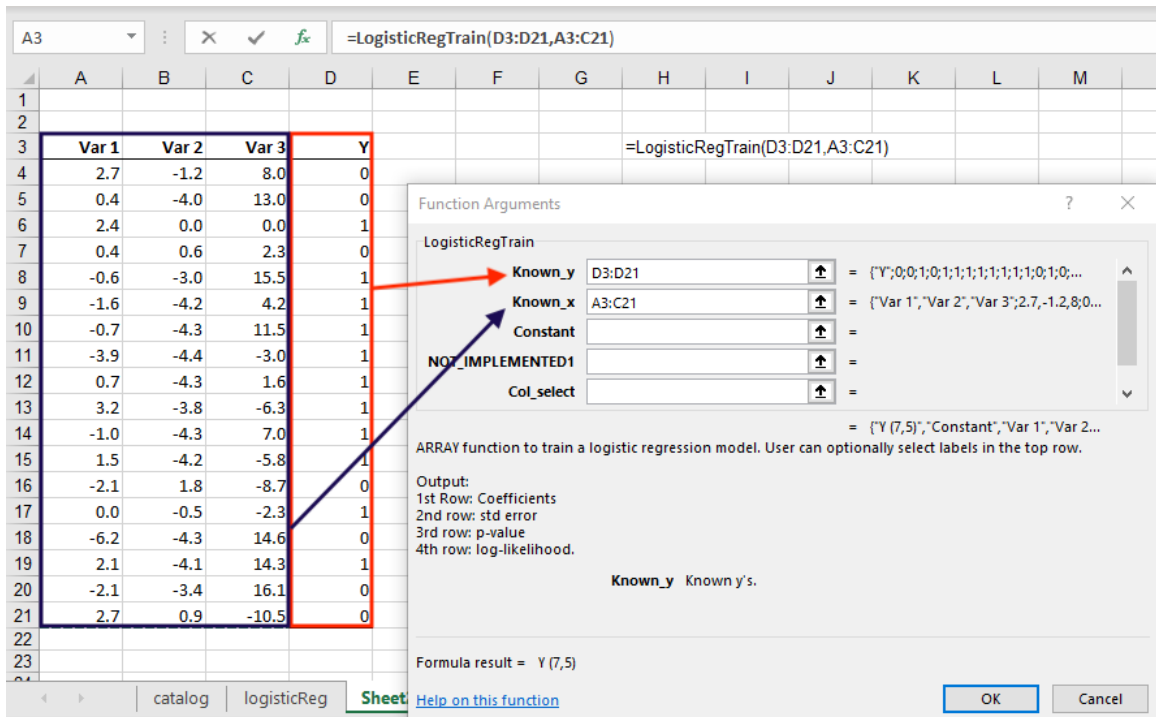
To use the function:

1. Select the cell where you want to display the output (H3 in the example) and call the function by typing **=LogisticRegTrain(** and then clicking on the f_x symbol next to the formula bar. (Alternatively, you can call the function by clicking on the f_x

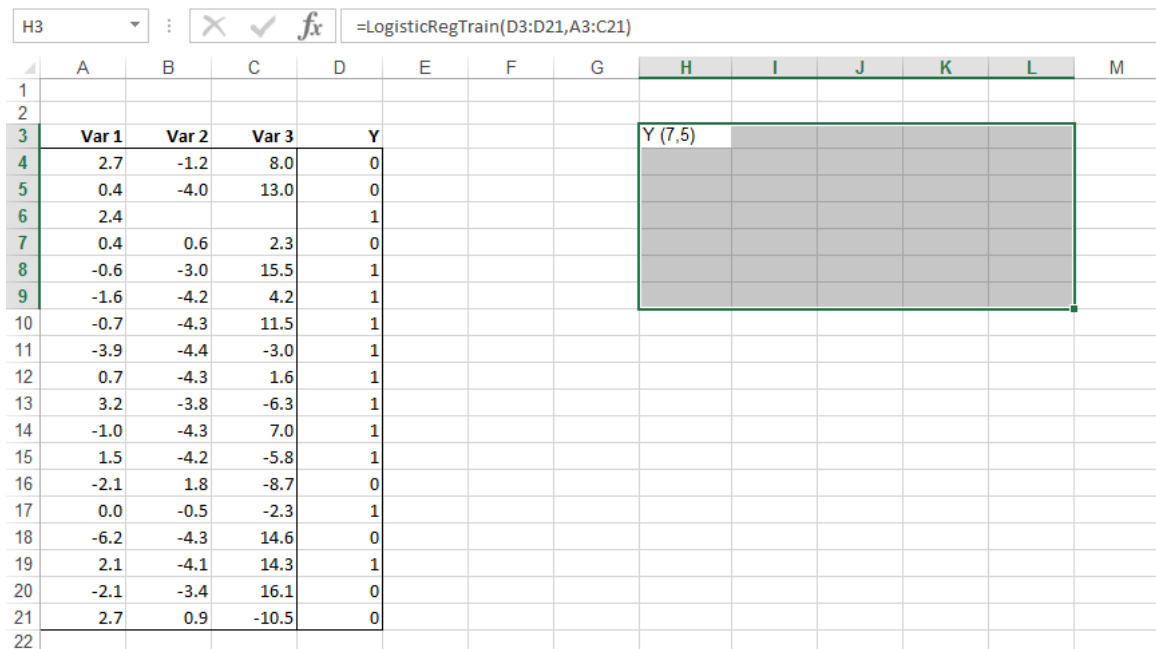
symbol first, which will display a menu of all available functions, and then clicking on LogisticRegTrain.)

SUM										
✕ ✓ f_x =LogisticRegTrain(
	A	B	C	D	E	F	G	H	I	J
1										
2										
3	Var 1	Var 2	Var 3	Y				=LogisticRegTrain(
4	2.7	-1.2	8.0	0						
5	0.4	-4.0	13.0	0						
6	2.4	0.0	0.0	1						
7	0.4	0.6	2.3	0						
8	-0.6	-3.0	15.5	1						
9	-1.6	-4.2	4.2	1						
10	-0.7	-4.3	11.5	1						
11	-3.9	-4.4	-3.0	1						
12	0.7	-4.3	1.6	1						
13	3.2	-3.8	-6.3	1						
14	-1.0	-4.3	7.0	1						
15	1.5	-4.2	-5.8	1						
16	-2.1	1.8	-8.7	0						
17	0.0	-0.5	-2.3	1						
18	-6.2	-4.3	14.6	0						
19	2.1	-4.1	14.3	1						
20	-2.1	-3.4	16.1	0						
21	2.7	0.9	-10.5	0						
22										

- When you call the function, the window **Function Arguments** will appear. Follow the instructions to populate the input fields, then click **OK**.



- The string Y(7,5) will appear in cell H3, where Y is the name of the variable in cell D3. This indicates that you should select an output array of 7 rows and 5 columns. Select the output array starting from cell H3 as the next figure shows.



- Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the output. *Remark: for recent versions of Excel, this step is not necessary.*

N11													
	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3	Var 1	Var 2	Var 3	Y				Y (7,5)	Constant	Var 1	Var 2	Var 3	
4	2.7	-1.2	8.0	0				Coefficients	-1.34837	0.367215	-1.21046	-0.18438	
5	0.4	-4.0	13.0	0				Std error	1.054453	0.314121	0.588221	0.124832	
6	2.4	0.0	0.0	1				p-value	20.10%	24.24%	3.96%	13.97%	
7	0.4	0.6	2.3	0				Log-likelihood	-7.36825				
8	-0.6	-3.0	15.5	1				Number valid obs	18				
9	-1.6	-4.2	4.2	1				Total obs	18				
10	-0.7	-4.3	11.5	1									
11	-3.9	-4.4	-3.0	1									
12	0.7	-4.3	1.6	1									
13	3.2	-3.8	-6.3	1									
14	-1.0	-4.3	7.0	1									
15	1.5	-4.2	-5.8	1									
16	-2.1	1.8	-8.7	0									
17	0.0	-0.5	-2.3	1									
18	-6.2	-4.3	14.6	0									
19	2.1	-4.1	14.3	1									
20	-2.1	-3.4	16.1	0									
21	2.7	0.9	-10.5	0									
22													

The second row of the output contains the estimates of the coefficients: the column (**Constant**) refers to the coefficient β_0 , while the columns (**Var1,Var2,Var3**) refer to the coefficients ($\beta_1, \beta_2, \beta_3$) respectively. In the following rows we have standard errors and p-values of the estimates. The last two rows contain the number of valid observations and the total number of observations respectively.

You can also format the output to display only three decimals and to display percentages.

Optional Input: Col_select. The optional input **Col_select** allows you to specify which variables you want to include in your regression. For example, the configuration below runs a logistic regression with only **Var1** and **Var3** as independent variables.

The screenshot shows an Excel spreadsheet with columns A through M. Column A contains values 1, 0, 1, 2.7, 0.4, 2.4, 0.4, -0.6, -1.6, -0.7, -3.9, 0.7, 3.2, -1.0, 1.5, -2.1, 0.0, -6.2, 2.1, -2.1, 2.7. Column B contains values 0, -1.2, -4.0, 0.0, 0.6, -3.0, -4.2, -4.4, -4.3, -4.3, -3.8, -4.3, -4.3, -4.3, -4.2, 1.8, -0.5, -4.3, -4.1, -3.4, 0.9. Column C contains values 1, 8.0, 13.0, 0.0, 2.3, 15.5, 4.2, -3.0, 11.5, -3.0, 1.6, -6.3, 7.0, -5.8, -8.7, 14.6, 14.3, 16.1, -10.5. Column D contains values 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0. Column E contains the text 'column select (1 include, 0 exclude)'. The dialog box for the LogisticRegTrain function is open, showing arguments: Known_y (D3:D21), Known_x (A3:C21), Constant (NOT_IMPLEMENTED1), and Col_select (A1:C1). The formula result is Y (7,4).

1.2.2. LogisticRegPredict

The example illustrated below is available on the add-in webpage by downloading the Linear and Logistic Regression file and going to the logisticReg sheet.

LogisticRegPredict uses the coefficients ($\beta_0^e, \beta_1^e, \dots, \beta_K^e$) that we estimate with **LogisticRegTrain** and the data on the independent variables (Var1, Var2, ..., VarK) to compute a prediction p for the probability that the dependent variable is equal to 1. The prediction is calculated in two steps:

- (i) Obtain the exponent w of the logistic function as

$$w = \beta_0^e + \beta_1^e \text{Var1} + \beta_2^e \text{Var2} + \dots + \beta_K^e \text{VarK}.$$

- (ii) Calculate the probability that $Y=1$ as follows

$$p = \frac{\exp(w)}{1 + \exp(w)}.$$

For example, using the coefficients that we estimated in the **LogisticRegTrain** example we would predict p for the first observation in the example as follows

$$w = -2.047 + 0.301 * 2.7 - 1.432 * (-1.2) - 0.207 * 8 = -1.172$$

and

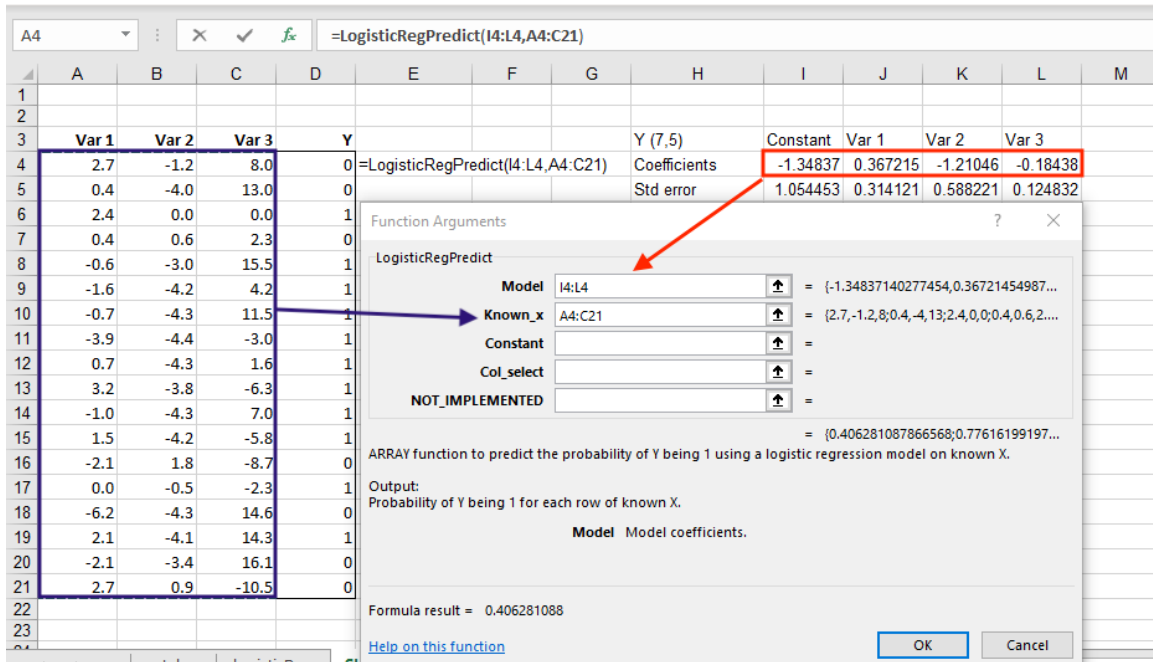
$$p = \frac{\exp(-1.172)}{1 + \exp(-1.172)} = 23.61\%$$

To use the function:




1. Select the first cell in which you want the output to be displayed; call the function by typing `=LogisticRegPredict(` in the formula bar and then pressing the f_x symbol next to the formula bar.

SUM													
X ✓ f_x =LogisticRegPredict(
	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3	Var 1	Var 2	Var 3	Y				Y (7,5)	Constant	Var 1	Var 2	Var 3	
4	2.7	-1.2	8.0	0	=LogisticRegPredict(Coefficients	-1.34837	0.367215	-1.21046	-0.18438	
5	0.4	-4.0	13.0	0				Std error	1.054453	0.314121	0.588221	0.124832	
6	2.4	0.0	0.0	1				p-value	20.10%	24.24%	3.96%	13.97%	
7	0.4	0.6	2.3	0				Log-likelihood	-7.36825				
8	-0.6	-3.0	15.5	1				Number valid obs	18				
9	-1.6	-4.2	4.2	1				Total obs	18				
10	-0.7	-4.3	11.5	1									
11	-3.9	-4.4	-3.0	1									
12	0.7	-4.3	1.6	1									
13	3.2	-3.8	-6.3	1									
14	-1.0	-4.3	7.0	1									
15	1.5	-4.2	-5.8	1									
16	-2.1	1.8	-8.7	0									
17	0.0	-0.5	-2.3	1									
18	-6.2	-4.3	14.6	0									
19	2.1	-4.1	14.3	1									
20	-2.1	-3.4	16.1	0									
21	2.7	0.9	-10.5	0									
22													

2. Follow the instructions to populate the input fields, as demonstrated in the figure below, then click OK.



- Remember that this is an array function. Select the column in which you want the output to be displayed: this should be a column of the same length as the number of observations that will contain the predictions associated to each observation. In the figure below, we have 18 observations and we select cells E4:E21.

E4	:				=LogisticRegPredict(I4:L4,A4:C21)									
	A	B	C	D	E	F	G	H	I	J	K	L	M	
1														
2														
3	Var 1	Var 2	Var 3	Y				Y (7,5)	Constant	Var 1	Var 2	Var 3		
4	2.7	-1.2	8.0	0	0.236122			Coefficients	-2.047	0.301	-1.432	-0.207		
5	0.4	-4.0	13.0	0				Std error	1.300	0.322	0.693	0.141		
6	2.4			1				p-value	11.53%	35.02%	3.86%	14.27%		
7	0.4	0.6	2.3	0				Log-likelihood	-6.159					
8	-0.6	-3.0	15.5	1				Number valid obs	17					
9	-1.6	-4.2	4.2	1				Total obs	18					
10	-0.7	-4.3	11.5	1										
11	-3.9	-4.4	-3.0	1										
12	0.7	-4.3	1.6	1										
13	3.2	-3.8	-6.3	1										
14	-1.0	-4.3	7.0	1										
15	1.5	-4.2	-5.8	1										
16	-2.1	1.8	-8.7	0										
17	0.0	-0.5	-2.3	1										
18	-6.2	-4.3	14.6	0										
19	2.1	-4.1	14.3	1										
20	-2.1	-3.4	16.1	0										
21	2.7	0.9	-10.5	0										
22														

- Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the predictions. You can also add a title (for example "LogisticRegPredict") to the output-column and format the output-cells to display percentages, as in the figure below. *Remark: for recent versions of Excel, this step is not necessary.*

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3	Var 1	Var 2	Var 3	Y				Y (7,5)	Constant	Var 1	Var 2	Var 3	
4	2.7	-1.2	8.0	0	0.406281088			Coefficients	-1.34837	0.367215	-1.21046	-0.18438	
5	0.4	-4.0	13.0	0	0.776161992			Std error	1.054453	0.314121	0.588221	0.124832	
6	2.4	0.0	0.0	1	0.385313171			p-value	20.10%	24.24%	3.96%	13.97%	
7	0.4	0.6	2.3	0	0.086920689			Log-likelihood	-7.36825				
8	-0.6	-3.0	15.5	1	0.311058582			Number valid obs	18				
9	-1.6	-4.2	4.2	1	0.91479336			Total obs	18				
10	-0.7	-4.3	11.5	1	0.814452492								
11	-3.9	-4.4	-3.0	1	0.956836256								
12	0.7	-4.3	1.6	1	0.9785141								
13	3.2	-3.8	-6.3	1	0.996271542								
14	-1.0	-4.3	7.0	1	0.900136079								
15	1.5	-4.2	-5.8	1	0.995301276								
16	-2.1	1.8	-8.7	0	0.063312717								
17	0.0	-0.5	-2.3	1	0.420900958								
18	-6.2	-4.3	14.6	0	0.247490071								
19	2.1	-4.1	14.3	1	0.851834057								
20	-2.1	-3.4	16.1	0	0.274394336								
21	2.7	0.9	-10.5	0	0.620025405								
22													

Optional Input: Col_select. See explanation at the end of Section 1.2.1.

1.3. K-Nearest Neighbor Functions (Content-based)

Tip: The functions in this section do content-based prediction, which means that they predict a score (or rating) based on other known attributes of content. The dependent variable (score or rating) and the independent variables (other attributes) are distinct, and in general you need information on both to be able to use these functions.

1.3.1. knn_in

The example illustrated below is available on the *add-in webpage* by downloading the *Classification, KNN, Misc file* and going to the *knn sheet*.

knn_in applies the **k**-nearest neighbor algorithm to make predictions within a specified sample. The inputs of the function are: the parameter **k** of nearest neighbor to match; the known variables (**Var1**, **Var2**, ..., **VarM**), on the basis of which the observations are matched; the known outcomes **Y**. The nearest neighbor to a given observation are the observations which have minimum Euclidean distance

$$d(i, j) = \sqrt{(\text{Var1}_i - \text{Var1}_j)^2 + (\text{Var2}_i - \text{Var2}_j)^2 + \dots + (\text{VarM}_i - \text{VarM}_j)^2}$$

and the prediction p is computed as a simple average of the **k**-nearest neighbor outcomes

$$p = \frac{Y_1 + Y_2 + \dots + Y_k}{k}.$$

To use the function:

1. Write the parameter **k** in an empty cell, **k=3** in the example below; select the first cell in which you want the output to be displayed; call the function by typing **=knn_in(** in the formula bar and then pressing the **f_x** symbol next to the formula bar.

SUM		X		✓		fx		=knn_in(
	A	B	C	D	E	F			
1									
2									
3			k		3				
4									
5	Var 1	Var 2	Y						
6		9.4	11.9	5	=knn_in(
7		10.8	10.2	5					
8		6.1	10	3					
9		11	4	3					
10		3.7	11.8	3					
11		11.4	2.9	3					
12		5.5	6	2					
13		5.1	2	1					
14		10.7	7.7	4					
15		2.5	10.4	3					
16		4.1	8.6	3					
17		7.3	5	2					
18		2.2	4.1	2					

2. Follow the instructions to populate the input fields, as demonstrated in the figure below, then click OK.

The screenshot shows the Excel KNN algorithm interface. The main window displays a data table with columns Var 1, Var 2, and Y. A red circle highlights the value 'k' in cell D3, which is the predicted Y value. A blue arrow points from the 'K' input in the Function Arguments dialog to the 'k' in the data table. The Function Arguments dialog shows Known_y as C6:C18, Known_x as A6:B18, and K as D3. The output is Y predictions.

- Remember that this is an array function. Select the column in which you want the output to be displayed: this should be a column of the same length as the number of observations. In the figure below, we selected cells D6:D18.

D6						
	A	B	C	D	E	F
1						
2						
3			k	3		
4						
5	Var 1	Var 2	Y			
6	9.4	11.9	5	4.00		
7	10.8	10.2	5			
8	6.1	10.0	3			
9	11.0	4.0	3			
10	3.7	11.8	3			
11	11.4	2.9	3			
12	5.5	6.0	2			
13	5.1	2.0	1			
14	10.7	7.7	4			
15	2.5	10.4	3			
16	4.1	8.6	3			
17	7.3	5.0	2			
18	2.2	4.1	2			
19						
20						

4. Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the predictions. You can also add a title (for example “knn_in predict”) to the output-column and format the output-cells to display only the first two decimals. *Remark: for recent versions of Excel, this step is not necessary.*

G6						
	A	B	C	D	E	F
1						
2						
3			k	3		
4						
5	Var 1	Var 2	Y			
6	9.4	11.9	5	4.00		
7	10.8	10.2	5	4.00		
8	6.1	10.0	3	3.00		
9	11.0	4.0	3	3.00		
10	3.7	11.8	3	3.00		
11	11.4	2.9	3	3.00		
12	5.5	6.0	2	2.33		
13	5.1	2.0	1	2.00		
14	10.7	7.7	4	3.33		
15	2.5	10.4	3	3.00		
16	4.1	8.6	3	2.67		
17	7.3	5.0	2	2.00		
18	2.2	4.1	2	2.00		
19						

1.3.2. knn_in_rmse

The example illustrated below is available on the add-in webpage by downloading the *Classification, KNN, Misc* file and going to the *knn_full* sheet.

knn_in_rmse is a function that computes the Root Mean Squared Error (RMSE) for k-nearest neighbor predictions (as the ones you can obtain with **knn_in**).

The inputs of the function are: the known variables (**Var1**, **Var2**, ..., **VarM**) on the basis of which the observations are matched; the known outcomes **Y**; the parameter **k** for which we want to compute the RMSE of predictions.

For a given **k**, the error is computed as follows

$$RMSE = \sqrt{\frac{(Y_1 - p_1)^2 + (Y_2 - p_2)^2 + \dots + (Y_N - p_N)^2}{N}},$$

where N is the total number of observations.

Note: this is not an array function and can be evaluated in a single stand-alone cell, in which case it outputs the RMSE for a specified k . However, it is particularly useful when you want to compare RMSEs for different k . We will illustrate how to do this in the example below.

To use the function:

1. Write a list of parameters k for which you want to compute the RMSE, $k = 1, 2, \dots, 10$ in the example below; select the cell next to the first value for k ; call the function by typing `=knn_in_rmse(` in the formula bar and then pressing the f_x symbol next to the formula bar.

	A	B	C	D	E	F	G	H
1								
2								
3	Var 1	Var 2	Y		k			
4	9.4	11.9	5		1	=knn_in_rmse(
5	10.8	10.2	5		2			
6	6.1	10	3		3			
7	11	4	3		4			
8	3.7	11.8	3		5			
9	11.4	2.9	3		6			
10	5.5	6	2		7			
11	5.1	2	1		8			
12	10.7	7.7	4		9			
13	2.5	10.4	3		10			
14	4.1	8.6	3					
15	7.3	5	2					
16	2.2	4.1	2					
17								
18								
19								

2. Input the references to the cells that contain the independent and dependent variable data, as in the figure below, then *lock* the cell references with \$ signs. Input the reference of the cell that contains the first value for k , and make sure that this is *not* locked. Then click OK.

Function Arguments

knn_in_rmse

Known_y: \$C\$4:\$C\$16

Known_x: \$A\$4:\$B\$16

Normalize: ☐ (selected)

NOT_IMPLEMENTED1: ☐ (selected)

Function to compute RMSE for output Y values computed using KNN algorithm.

Known_x: Known X values.

Formula result = 0.480384461

OK Cancel

3. The function will display the output in the selected cell (F4). Copy the content of the cell down until cell F13, the formula will automatically update the calculation for every k . You can also format the output to display only three decimals.

F4								
	A	B	C	D	E	F	G	H
1								
2								
3	Var 1	Var 2	Y		k			
4	9.4	11.9	5		1	0.480		
5	10.8	10.2	5		2	0.519		
6	6.1	10	3		3	0.531		
7	11	4	3		4	0.676		
8	3.7	11.8	3		5	0.779		
9	11.4	2.9	3		6	0.859		
10	5.5	6	2		7	0.924		
11	5.1	2	1		8	0.956		
12	10.7	7.7	4		9	1.008		
13	2.5	10.4	3		10	1.026		
14	4.1	8.6	3					
15	7.3	5	2					
16	2.2	4.1	2					
17								

1.3.3. knn_out

The example illustrated below is available on the add-in webpage by downloading the *Classification, KNN, Misc* file and going to the *knn_full* sheet.

knn_out applies the-k nearest neighbor algorithm to make predictions on the outcome variable Y of a new set of observations using a known sample.

To use the function:

1. Write the parameter **k** in an empty cell, **k=3** in the example below. Select the first cell in which you want the output to be displayed; call the function by typing **=knn_out(** in the formula bar and then pressing the f_x symbol next to the formula bar.

SUM													:	X	✓	f_x	=knn_out(
1	A	B	C	D	E	F	G	H	I	J	K	L					
2																	
3			k	3		Out-of-sample predictions											
4						(Use the model based on 13 obserations to predict 6 new observations)											
5	Var 1	Var 2	Y			Var 1	Var 2	=knn_out(
6	9.4	11.9	5			9.3	2.7										
7	10.8	10.2	5			5.3	10.2										
8	6.1	10	3			11.9	3.4										
9	11	4	3			4.1	2.2										
10	3.7	11.8	3			6.6	4.3										
11	11.4	2.9	3			8.3	5.4										
12	5.5	6	2														
13	5.1	2	1														
14	10.7	7.7	4														
15	2.5	10.4	3														
16	4.1	8.6	3														
17	7.3	5	2														
18	2.2	4.1	2														
19																	

2. The function takes as inputs the new observations **X**, the old observations and the known outcomes, as well as the parameter **k**. Follow the instructions to populate the input fields, as demonstrated in the figure below, then click **OK**.

Excel spreadsheet showing the setup for the KNN algorithm. The formula bar displays `=knn_out(F6:G11,C6:C18,A6:B18,D3)`. The spreadsheet contains data for 13 observations (rows 6-18) with columns Var 1, Var 2, and Y. A new observation (row 19) is being predicted. The function arguments dialog box is open, showing the following arguments:

- Known_y**: F6:G11 (Range)
- Known_x**: C6:C18 (Range)
- Known_x**: A6:B18 (Range)
- Normalize**: K (Value)

The formula result is 2.666666667.

- Remember that this is an array function. Select the column in which you want the output to be displayed: this should be a column of the same length as the number of new observations.

Excel spreadsheet showing the setup for the KNN algorithm. The formula bar displays `=knn_out(E6:F11,C6:C18,A6:B18,C3)`. The spreadsheet contains data for 13 observations (rows 6-18) with columns Var 1, Var 2, and Y. A new observation (row 19) is being predicted. The function arguments dialog box is open, showing the following arguments:

- Known_y**: E6:F11 (Range)
- Known_x**: C6:C18 (Range)
- Known_x**: A6:B18 (Range)
- Normalize**: C3 (Value)

The formula result is 2.666666667.

- Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the predictions. You can also format the output to display only two decimals. *Remark: for recent versions of Excel, this step is not necessary.*

Excel spreadsheet showing the setup for the KNN algorithm. The formula bar displays `=knn_out(F6:G11,C6:C18,A6:B18,D3)`. The spreadsheet contains data for 13 observations (rows 6-18) with columns Var 1, Var 2, and Y. A new observation (row 19) is being predicted. The function arguments dialog box is open, showing the following arguments:

- Known_y**: F6:G11 (Range)
- Known_x**: C6:C18 (Range)
- Known_x**: A6:B18 (Range)
- Normalize**: D3 (Value)

The formula result is 2.666666667.

1.4. K-Nearest Neighbor Functions (User-based)

Tip: The functions in this section do user-based prediction, which means that they predict a score (or rating) based on the scores of other users. These function do not use any other content-attribute (as movie characteristics). You should use these functions when you want to predict ratings (for example, movie ratings) based only on the ratings of other users.

1.4.1. knn_in_movie

The example illustrated below is available on the [add-in webpage](#) by downloading the *Classification, KNN, Misc* file and going to the *knn_in_movie* sheet.

knn_in_movie uses the **k**-nearest neighbor algorithm to make predictions within a specified sample of users' ratings.

To use the function:

1. Write the parameter **k** in an empty cell, **k=3** in the example below; select the first cell in which you want the output to be displayed; call the function by typing **=knn_in_movie(** in the formula bar and then pressing the **f_x** symbol next to the formula bar.

SUM	:	X	✓	f_x	=knn_in_movie(
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q				
1																					
2																					
3							k		3												
4																					
5	Movie rating data	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6				Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6					
6	User	User	Godfathe		Bourne	Pretty	Working	Moneyba			Godfathe		Bourne	Pretty	Working	Moneyba					
7	number	name	r	Top Gun	Identity	Woman	Girl	Il			r	Top Gun	Identity	Woman	Girl	Il					
8	1	Paul	5	2		1	0				Paul	=knn_in_movie(
9	2	Ben	5		5			2			Ben										
10	3	Jane	4	2		2	5	3			Jane										
11	4	Guiga		3	3	3	3	4			Guiga										
12	5	Thomas	4	4		4					Thomas										
13	6	Marisa	4	3	2			4			Marisa										
14	7	Lucas	1	0	3		3	2			Lucas										
15	8	Emma	4	5	4	5		4			Emma										
16	9	Olivia	5	2	2	2					Olivia										
17	10	Lewis	4	3		3		0			Lewis										

2. Follow the instructions to populate the input fields, as demonstrated in the figure below, then click OK.

Excel screenshot showing the formula bar with `=knn_in_movie(C7:H16,H3)` and the Function Arguments dialog box. The dialog box shows the Known_x argument as `K H3` and the K argument as `3`. The formula result is `4.666666667`.

User number	User name	Movie 1 Godfather	Movie 2 Top Gun	Movie 3 Bourne Identity	Movie 4 Pretty Woman	Movie 5 Working Girl	Movie 6 Moneyball
1	Paul	5	2		1	0	
2	Ben	5		5			2
3	Jane	4	2		2	5	3
4	Guiga		3	3	3	3	4
5	Thomas	4	4		4		
6	Marisa	4	3	2			4
7	Lucas		1	0	3	3	2
8	Emma	4	5	4	5		4
9	Olivia	5	2	2	2		
10	Lewis	4	3		3		0

- Remember that this is an array function. Select the area in which you want the output to be displayed: this should have the same number of rows and columns as the area that contains the users' ratings. In the figure below, we selected cells K7:P16.

Excel screenshot showing the formula bar with `=knn_in_movie(C7:H16,H3)` and the output area K7:P16. The output area is highlighted in green, showing the predicted rating for Paul (4.67) and the predicted ratings for the other users.

User number	User name	Movie 1 Godfather	Movie 2 Top Gun	Movie 3 Bourne Identity	Movie 4 Pretty Woman	Movie 5 Working Girl	Movie 6 Moneyball
1	Paul	5	2		1	0	
2	Ben	5		5			2
3	Jane	4	2		2	5	3
4	Guiga		3	3	3	3	4
5	Thomas	4	4		4		
6	Marisa	4	3	2			4
7	Lucas		1	0	3	3	2
8	Emma	4	5	4	5		4
9	Olivia	5	2	2	2		
10	Lewis	4	3		3		0

- Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the predictions. You can also add a title to the output-column and format the output-cells to display only the first two decimals. *Remark: for recent versions of Excel, this step is not necessary.*

Excel screenshot showing the formula bar with `=knn_in_movie(C7:H16,H3)` and the output area K7:P16. The output area is highlighted in green, showing the predicted ratings for all users.

User number	User name	Movie 1 Godfather	Movie 2 Top Gun	Movie 3 Bourne Identity	Movie 4 Pretty Woman	Movie 5 Working Girl	Movie 6 Moneyball
1	Paul	5	2		1	0	
2	Ben	5		5			2
3	Jane	4	2		2	5	3
4	Guiga		3	3	3	3	4
5	Thomas	4	4		4		
6	Marisa	4	3	2			4
7	Lucas		1	0	3	3	2
8	Emma	4	5	4	5		4
9	Olivia	5	2	2	2		
10	Lewis	4	3		3		0

1.4.2. knn_in_movie_rmse

The example illustrated below is available on the *add-in webpage* by downloading the *Classification, KNN, Misc* file and going to the *knn_in_movie* sheet.

knn_in_movie_rmse is a function that computes the Root Mean Squared Error (RMSE) for k-nearest neighbor predictions (as the ones you can obtain with **knn_in_movie**).

The inputs of the function are the users' ratings for a set of movies and the parameter **k** for which we want to compute the RMSE of predictions.

Note: this is not an array function and can be evaluated in a single stand-alone cell, in which case it outputs the RMSE for a specified k. However, it is particularly useful when you want to compare RMSEs for different k. We will illustrate how to do this in the example below.

To use the function:

1. Write a list of parameters **k** for which you want to compute the RMSE, $k = 1, 2, \dots, 9$ in the example below; select the cell next to the first value for **k**; call the function by typing **=knn_in_rmse(** in the formula bar and then pressing the **f_x** symbol next to the formula bar.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3		Movie rating data	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6					
4		User number	User name	Godfathe	Top Gun	Bourne Identity	Pretty Woman	Working Girl	Moneyba	II	k		
5		1	Paul	5	2		1	0			1	=knn_in_movie_rmse(
6		2	Ben	5		5			2		2		
7		3	Jane	4	2		2	5	3		3		
8		4	Guiga		3	3	3	3	4		4		
9		5	Thomas	4	4		4				5		
10		6	Marisa	4	3	2			4		6		
11		7	Lucas		1	0	3	3	2		7		
12		8	Emma	4	5	4	5		4		8		
13		9	Olivia	5	2	2	2				9		
14		10	Lewis	4	3		3		0				
15													

2. Input the references to the cells that contain the users' ratings, as in the figure below, then *lock* the cell references with \$ signs. Input the reference of the cell that contains the first value for **k**, and make sure that this is *not locked*. Then click OK.

Function Arguments

knn_in_movie_rmse

Known_x: \$C\$5:\$H\$14

Normalize: 1

NOT_IMPLEMENTED1: 1

Weighted_voting: 1

Function to compute RMSE for in_movie ratings computed using KNN_in_movie algorithm.

Known_x: Known X values.

Formula result = 1.112697281

Help on this function

OK Cancel

3. The function will display the output in the selected cell (K5). Copy the content of the cell down until cell K13, the formula will automatically update the calculation for every k. You can also format the output to display only three decimals.

[illegible]

1.5. Other KNN Functions

The example illustrated below is available on the `add-in` webpage by downloading the `Classification`, `KNN`, `Misc` file and going to the `knn_full` sheet.

1.5.1. knn_dist

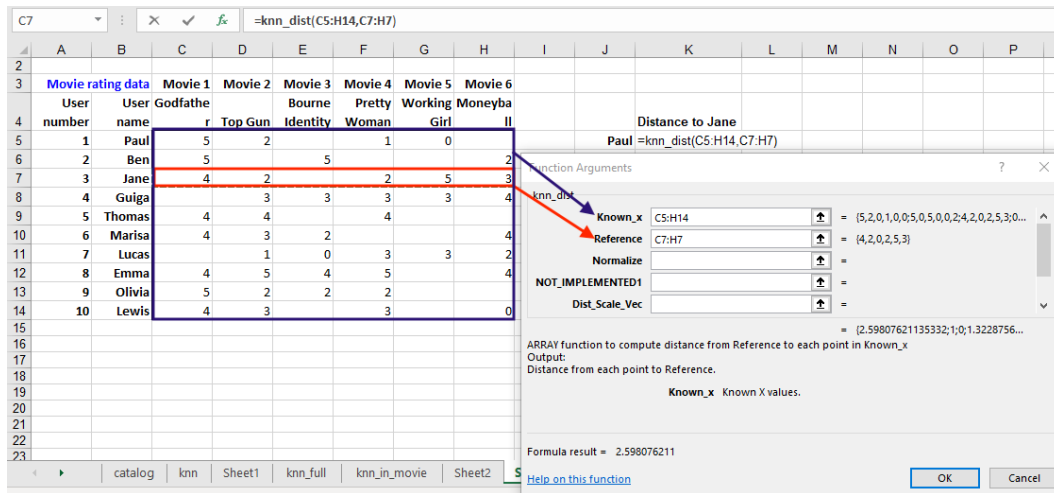
knn_dist is an array function that uses users' ratings to compute the distance of each user from a given user that you specify.

To use the function:

0. Decide the baseline user from which you want to compute the distances (we use **Jane** in the example below).
1. Select the first cell in which you want the output to be displayed; call the function by typing `=knn_dist(` in the formula bar and then pressing the $\mathbf{f_x}$ symbol next to the formula bar.

[illegible]

- Follow the instructions to populate the input fields, as demonstrated in the figure below, then click OK.



- Remember that this is an array function. Select the column in which you want the output to be displayed: this should be a column of the same length as the number of users. In the figure below, we selected cells K5:K14.

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3		Movie rating data	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6				
4		User number	User name	Godfather	Top Gun	Bourne Identity	Pretty Woman	Working Girl	Moneyball			
5		1	Paul	5	2		1				Paul	0.82
6		2	Ben	5		5			2		Ben	
7		3	Jane	4	2		2	5	3		Jane	
8		4	Guiga		3	3	3	3	4		Guiga	
9		5	Thomas	4	4		4				Thomas	
10		6	Marisa	4	3	2			4		Marisa	
11		7	Lucas		1	0	3	3	2		Lucas	
12		8	Emma	4	5	4	5		4		Emma	
13		9	Olivia	5	2	2	2				Olivia	
14		10	Lewis	4	3		3				Lewis	

- Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the predictions. You can also add a title to the output-column and format the output-cells to display only the first two decimals. *Remark: for recent versions of Excel, this step is not necessary.*

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3		Movie rating data	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6				
4		User number	User name	Godfather	Top Gun	Bourne Identity	Pretty Woman	Working Girl	Moneyball			
5		1	Paul	5	2		1	0			Paul	2.60
6		2	Ben	5		5			2		Ben	1.00
7		3	Jane	4	2		2	5	3		Jane	0.00
8		4	Guiga		3	3	3	3	4		Guiga	1.32
9		5	Thomas	4	4		4				Thomas	1.63
10		6	Marisa	4	3	2			4		Marisa	0.82
11		7	Lucas		1	0	3	3	2		Lucas	1.32
12		8	Emma	4	5	4	5		4		Emma	2.18
13		9	Olivia	5	2	2	2				Olivia	0.58
14		10	Lewis	4	3		3		0		Lewis	1.66

1.5.2. knn_nearest

knn_nearest is an array function that uses users' ratings to rank users in order of proximity (smallest distance) to a given user that you specify. The numbers returned are indices of the rows in the input table in order of distance. (Note, in particular, that they are not “ranks”.)

To use the function:

0. Decide the baseline user from which you want to compute the distance rankings (we use **Jane** in the example below).
1. Select the first cell in which you want the output to be displayed; call the function by typing `=knn_nearest(` in the formula bar and then pressing the **f_x** symbol next to the formula bar.

	A	B	C	D	E	F	G	H	I	J	K	L
2												
3		Movie rating data	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6				
4		User number	User name	Godfathe	Bourne Identity	Pretty Woman	Working Moneyba	Girl II				
5	1	Paul	5	2		1	0					
6	2	Ben	5		5			2				
7	3	Jane	4	2		2	5	3				
8	4	Guiga		3	3	3	3	4				
9	5	Thomas	4	4		4						
10	6	Marisa	4	3	2			4				
11	7	Lucas		1	0	3	3	2				
12	8	Emma	4	5	4	5		4				
13	9	Olivia	5	2	2	2						
14	10	Lewis	4	3		3		0				
15												

2. Follow the instructions to populate the input fields, as demonstrated in the figure below, then click OK.

The screenshot shows the Excel interface with the `=knn_nearest` function dialog box open. The dialog box has the following fields:

- Known_x:** C5:H14
- Reference:** C7:H7
- Normalize:** (empty)
- NOT_IMPLEMENTED1:** (empty)
- Dist_Scale_Vec:** (empty)

The formula bar shows `=knn_nearest(C5:H14,C7:H7)`. The spreadsheet shows the same data as the previous table, with rows 5-14 highlighted in blue.

3. Remember that this is an array function. Select the column in which you want the output to be displayed: this should be a column of the same length as the number of users. In the figure below, we selected cells K5:K14.

Actual	Predicted
8.0	8.3
9.3	9.1
7.5	7.2
8.9	10.0
10.2	10.1
8.3	8.3
8.8	9.0
8.8	8.7
10.7	10.6
11.7	11.2

We illustrate in this section a function that computes the Root Mean Square Error.

To use the function:

1. Select the cell in which you want the output to be displayed; call the function by typing `=BA_RMSE(` in the formula bar and then pressing the f_x symbol next to the formula bar.

	A	B	C	D	E	F	G	H	I
1									
2									
3	Actual	Predicted							
4	8.0	8.3			=BA_RMSE(
5	9.3	9.1							
6	7.5	7.2							
7	8.9	10.0							
8	10.2	10.1							
9	8.3	8.3							
10	8.8	9.0							
11	8.8	8.7							
12	10.7	10.6							
13	11.7	11.2							
14									
15									
16									

2. Follow the instructions to populate the inputs of the function.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2													
3	Actual	Predicted											
4	8.0	8.3											
5	9.3	9.1											
6	7.5	7.2											
7	8.9	10.0											
8	10.2	10.1											
9	8.3	8.3											
10	8.8	9.0											
11	8.8	8.7											
12	10.7	10.6											
13	11.7	11.2											
14													
15													
16													
17													
18													
19													
20													

Function Arguments

BA_RMSE

Predicted B4:B13 = {8.3;9.1;7.2;10.1;8.3;9.0;8.7;10.6;10.7;11.2}

Actual A4:A13 = {8.0;9.3;7.5;8.9;10.2;8.3;8.8;8.8;10.7;11.7}

NOT_IMPLEMENTED =

Function to compute the RMSE between predicted and actual values.

Predicted Column containing predictions.

Formula result = 0.418330086

[Help on this function](#) OK Cancel

3. Click OK to display the result, in the example we have **RMSE=0.42**.

1.7. Classification Functions

In this section we will illustrate a set of functions that you can use to solve classification problems. All the examples will be built upon the following dataset. Suppose that you have access to 17 experiments (e.g. clinical tests, spam/non-spam email classification, etc.) for which you observe the **actual outcome** of the experiment (1 if successful and 0 if not successful) and the **probability of success** generated by some predictive model.

	Predicted
Actual	probability
outcome	(score)
0	0.04
0	0.11
0	0.62
0	0.40
1	0.62
1	0.76
1	0.38
1	0.61
0	0.22
1	0.36
0	0.29
1	0.20
0	0.18
1	0.11
1	0.30
1	0.31
0	0.18

1.7.1. confusionMatrix

*The example illustrated below is available on the [add-in webpage](#) by downloading the *Classification, KNN, Misc file* and going to the *ROC_no_cost* sheet.*

Before illustrating the function, we build a classifier on the above dataset with a classification threshold of 0.6: we predict as successful (1) the experiments that have a probability of success of at least 0.6 and not successful (0) the others.

	Predicted	Classification
Actual	probability	decision
outcome	(score)	(threshold 0.6)
0	0.04	0
0	0.11	0
0	0.62	1
0	0.40	0
1	0.62	1
1	0.76	1
1	0.38	0
1	0.61	1
0	0.22	0
1	0.36	0
0	0.29	0
1	0.20	0
0	0.18	0
1	0.11	0
1	0.30	0
1	0.31	0
0	0.18	0

We will use **confusionMatrix** to compute the Confusion Matrix associated to the above classifier.

To use the function:

1. Select the first cell in which you want the output to be displayed; call the function by typing **=confusionMatrix(** in the formula bar and then pressing the f_x symbol next to the formula bar.
2. Follow the instructions to populate the input fields, as demonstrated in the figure below, then click OK. (The actual and predicted values must both be binary.)

The screenshot shows an Excel spreadsheet with the following data in columns A, B, and C:

Actual outcome	Predicted probability (score)	Classification decision (threshold 0.6)
0	0.04	0
0	0.11	0
0	0.62	1
0	0.40	0
1	0.62	1
1	0.76	1
1	0.38	0
1	0.61	1
0	0.22	0
1	0.36	0
0	0.29	0
1	0.20	0
0	0.18	0
1	0.11	0
1	0.30	0
1	0.31	0
0	0.18	0

The formula bar shows the function: **=confusionMatrix(A6:A22,C6:C22)**

The 'Function Arguments' dialog box is open, showing the following inputs:

- Actual:** A6:A22
- Prediction:** C6:C22

The dialog box also displays the resulting confusion matrix and the formula result:

Formula result = Confusion matrix (5,4)

- Remember that this is an array function. Select the set of cells in which you want the output to be displayed. In the figure below, we selected cells E3:H7. *Remark: for recent versions of Excel, this step is not necessary.*

The screenshot shows an Excel spreadsheet with the following data in columns A through D:

	A	B	C
1			
2			
3		Predicted	Classification
4	Actual	probability	decision
5	outcome	(score)	(threshold 0.6)
6	0	0.04	0
7	0	0.11	0
8	0	0.62	1
9	0	0.40	0
10	1	0.62	1
11	1	0.76	1
12	1	0.38	0
13	1	0.61	1
14	0	0.22	0
15	1	0.36	0
16	0	0.29	0
17	1	0.20	0
18	0	0.18	0
19	1	0.11	0
20	1	0.30	0
21	1	0.31	0
22	0	0.18	0
23			

The formula bar shows the array formula: `=confusionMatrix(A6:A22,C6:C22)`. A range of cells E3:H7 is selected, and a tooltip indicates the output is a "Confusion matrix (5,4)".

- Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the output. You can also format the output-cells to display the labels of the matrix in boldface.

Confusion matrix (5,4)		Predicted	
Actual		1	0
	1	3	6
	0	1	7
Total		4	13

1.7.2. ROC

The example illustrated below is available on the *add-in* webpage by downloading the *Classification, KNN, Misc* file and going to the *ROC_no_cost* sheet.

An important measure for evaluating the performance of a given classification model is the error rate. For example, consider the confusion matrix calculated in the previous section: the classifier built using a threshold of 0.6 has a *false positive rate* of $1/8 = 12.5\%$ and a *true positive rate* of $3/9 = 33.3\%$. The ROC curve is simply a plot of the true positive versus the false positive rate for different thresholds. Note that, in principle, one can use any score for classification purposes. In the example, the score used was the predicted probability, but any other score can be used as well. By varying the classification threshold, different (FPR, TPR) pairs result, and the ROC curve captures all possible achievable pairs using that particular score/classifier. The ROC curve is a useful tool when you are trying to decide which classification threshold you should select.

To use the function:

0. Insert in a column the list of threshold values you want to test.

Probability	False	True
threshold	positive	positive
0.0		
0.1		
0.2		
0.3		
0.4		
0.5		
0.6		
0.7		
0.8		
0.9		
1.0		

1. Select the cell immediately to the right of the first threshold value, and call the function by typing `=ROC(` in the formula bar and then pressing the f_x symbol next to the formula bar.
2. Follow the instructions to populate the input fields, as demonstrated in the figure below, then click OK. (For the ROC curve with “Cost” see section 1.7.4 below.)

The screenshot shows an Excel spreadsheet with the following data:

Actual outcome	Predicted probability (score)	Probability threshold
0	0.04	0.0
0	0.11	0.1
0	0.62	0.2
0	0.40	0.3
1	0.62	0.4
1	0.76	0.5
1	0.38	0.6
1	0.61	0.7
0	0.22	0.8
1	0.36	0.9
0	0.29	1.0
1	0.20	
0	0.18	
1	0.11	
1	0.30	
1	0.31	
0	0.18	

The dialog box for the `ROC` function is open, showing the following arguments:

- Score:** B6:B22
- Actual:** A6:A22
- Threshold:** D6:D16
- Cost:**
- Reversed:**

The formula result is 1.

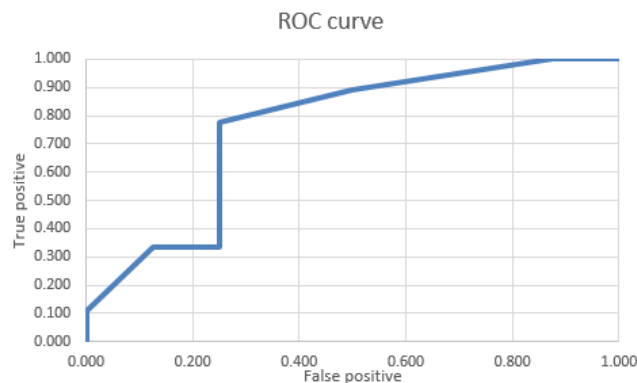
3. Remember that this is an array function. Select the set of cells in which you want the output to be displayed.

Probability	False	True
threshold	positive	positive
0.0	1.000	
0.1		
0.2		
0.3		
0.4		
0.5		
0.6		
0.7		
0.8		
0.9		
1.0		

4. Position the cursor in the formula bar, and press **Ctrl+Shift+Enter** to display the output. You can also format the output-cells to display only three decimals. *Remark: for recent versions of Excel, this step is not necessary.*

Probability	False	True
threshold	positive	positive
0.0	1.000	1.000
0.1	0.875	1.000
0.2	0.500	0.889
0.3	0.250	0.778
0.4	0.250	0.333
0.5	0.125	0.333
0.6	0.125	0.333
0.7	0.000	0.111
0.8	0.000	0.000
0.9	0.000	0.000
1.0	0.000	0.000

5. You can easily plot the false positive and true positive rates to display the ROC curve. To generate the graph below: select the set of cells that display the output of the **ROC** function; then select **INSERT - Charts - Scatter - Scatter with straight lines**.



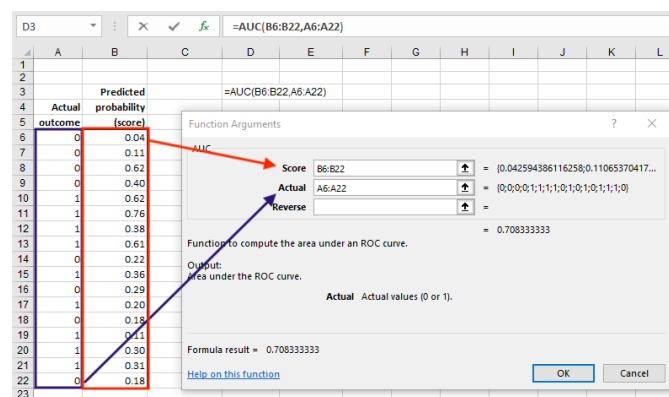
1.7.3. AUC

The example illustrated below is available on the add-in webpage by downloading the Classification, KNN, Misc file and going to the ROC_no_cost sheet.

This function computes the area under the ROC curve.

To use the function:

1. Select the cell in which you want the output to be displayed; call the function by typing `=AUC(` in the formula bar and then pressing the f_x symbol next to the formula bar.
2. Follow the instructions to populate the inputs of the function.



3. Click OK to display the result, in the example we have **AUC=0.71**.

1.7.4. ROC (with cost information)

The example illustrated below is available on the add-in webpage by downloading the Classification, KNN, Misc file and going to the ROC_with_cost sheet.

This is a variation on the basic **ROC** function, when you have a cost associated to each type of classification error. The costs must be expressed in terms of a cost matrix.

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3		Predicted									
4	Actual	probability		Probability	False	True	Total		Cost matrix	Predicted	
5	outcome	(score)		threshold	positive	positive	Cost		Actual	1	0
6	0	0.04		0.0					1	6	10
7	0	0.11		0.1					0	3	0
8	0	0.62		0.2							
9	0	0.40		0.3							
10	1	0.62		0.4							
11	1	0.76		0.5							
12	1	0.38		0.6							
13	1	0.61		0.7							
14	0	0.22		0.8							
15	1	0.36		0.9							
16	0	0.29		1.0							
17	1	0.20									
18	0	0.18									
19	1	0.11									
20	1	0.30									
21	1	0.31									
22	0	0.18									

To use the function follow the same steps as the basic **ROC** function, with the following variations:

- At *Step 2*. insert also the cost matrix information: type J6:K7 in the **Cost** input field.
- At *Step 3*. select three columns, as the function now outputs also the total cost.
Remark: for recent versions of Excel, this step is not necessary.

Probability	False	True	Total
threshold	positive	positive	Cost
0.0	1.000		
0.1			
0.2			
0.3			
0.4			
0.5			
0.6			
0.7			
0.8			
0.9			
1.0			

1.8. Monte Carlo Simulation

In this section we will illustrate how to run a Monte Carlo Simulation in a spreadsheet.

The example illustrated below is available on the add-in webpage by downloading the Monte Carlo Simulation file and going to the MCSim sheet.

Example: simple revenue simulation. Suppose that you have 100 customers coming into your shop on each day and that each customer purchases something with probability 40%. The accrued revenue per purchase is normally distributed with a mean of \$20 and a

standard deviation of \$5. Simulate the **number of purchases**, the **revenue per purchase**, and the **total revenues**. Run the simulation for 500 trials.

To set up the simulation:

S0. Populate the spreadsheet with the given data.

	A	B	C	D	E	F
1						
2						
3						
4	Simple revenue simulation					
5	Number of customers	100				
6	Probability of purchase	40%	Simulation trials	500		
7	Number of purchases (binomial)					
8						
9	Mean revenue per purchase	20				
10	Std dev of revenue per purchase	5				
11	Revenue per purchase (normal)					
12						
13	Total revenue					
14						

S1. Insert the formulas for the quantities you want to simulate.

(S1.i) Number of purchases: this is a random sample from a Binomial distribution with 100 trials and probability equal to 0.4. You can use the add-in function **BinomSim** to generate this number.

The screenshot shows the Excel spreadsheet with the **BinomSim** function dialog box open. The dialog box is titled "Function Arguments" and shows the following arguments:

- N**: 100 (from cell B5)
- P**: 0.4 (from cell B6)

The dialog box also displays the formula result as 44. The spreadsheet shows the formula **=BinomSim(B5,B6)** entered in cell B7.

(S1.ii) Revenue per purchase: this is a random sample from a Normal distribution with mean 20 and standard deviation 5. You can use the add-in function **NormalSim** to generate this number.

The screenshot shows the Excel spreadsheet with the **NormalSim** function dialog box open. The dialog box is titled "Function Arguments" and shows the following arguments:

- Mean**: 20 (from cell B9)
- Standard_dev**: 5 (from cell B10)

The dialog box also displays the formula result as 22.3. The spreadsheet shows the formula **=NormalSim(B9,B10)** entered in cell B11.

(S1.ii) Total Revenues: this is simply the product of number of purchases times revenue per purchase. Select cell B13 and insert the formula =B7*B11.

S2. Create the formula row, which will be your input for the simulation. Simply copy and paste the formulas from (1.i), (1.ii) and (1.iii) to an empty row. We did this in the figure below, in particular we inserted =B7 in cell E5, =B11 in cell F5, and =B13 in cell G5.

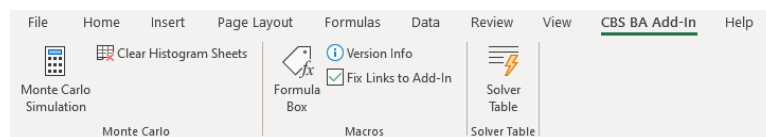
E5	:				=B7				
	A	B	C	D	E	F	G	H	
1									
2									
3									
4	Simple revenue simulation				Num purchase	Rev per purchase	Total rev		
5	Number of customers	100		Formula	32	21.8	698.9		
6	Probability of purchase	40%		Simulation trials	500				
7	Number of purchases (binomial)	32							
8									
9	Mean revenue per purchase	20							
10	Std dev of revenue per purchase	5							
11	Revenue per purchase (normal)	21.8							
12									
13	Total revenue	698.9							
14									

S3. You can also add an (optional) row of labels which indicates the variable for which you would like an histogram to be automatically generated, along with the simulation results. In our example we insert a row of labels with 0s for the first two variables and 1 for **total revenues**. As a result an histogram for this last variable will be generated, along with the simulation results.

		Num purchase	Rev per purchase	Total rev
	Formula	32	21.8	698.9
	Simulation trials	500		

To run the simulation:

R0. Go to the CBS BA ADD-IN tab and then select the Monte Carlo Simulation to pop up the MonteCarlo Simulation Dialog. (Alternatively, the dialog can be popped up by clicking Ctrl+Shift+M.)



R1. Populate the inputs of the function in the Dialog Box and then click OK. Also, by selecting the **Print Histogram** check box you can automatically generate histograms of the selected variables. Note that the simulation can track any number of variables in a single row, not just the three used in this example.

