

IE 5390 – Assignment - 09

Name: Simran Abhay Sinha

Lab 17

1. Use the cars dataset that we worked with in class. But this time, instead of using the `lm()` function for linear regression, create the linear regression model from scratch. You have the math preliminaries to help you with the model components. Some hints on what you will need to do:

- i) Establish what x and y are. In this case, x is speed and y is the stopping distance.
- ii) Calculate S_{xx} , S_{xy} and S_{yy}
- iii) Then calculate β_0 and β_1

Loading the dataset

Cars is built-in dataset

```
5 > ```{r}
6 # Lab 17_1
7 data(cars) # Load built-in dataset
8 x <- cars$speed # Independent variable (speed)
9 y <- cars$dist # Dependent variable (stopping distance)
10 > ```
11
```

Answer the following questions:

a) What are the values for Sxx, Sxy and Syy that you obtained?

Code:

```
12 = ```{r}
13 #Lab 17_A
14 x_mean <- mean(x)
15 y_mean <- mean(y)
16
17 Sxx <- sum((x - x_mean)^2)
18 Sxy <- sum((x - x_mean) * (y - y_mean))
19 Syy <- sum((y - y_mean)^2)
20
21 Sxx
22 Sxy
23 Syy
24 = ```
```

Output:

The screenshot shows an RStudio interface with the following components:

- Source Editor:** Contains the R code from the previous block, with line numbers 12 through 24.
- Console:** Displays the output of the code:


```
[1] 1778
[1] 3387.4
[1] 6254.96
```
- Environment:** Shows the values of the variables created:

Variable	Value
x_mean	1778
y_mean	3387.4
Sxx	1778
Sxy	3387.4
Syy	6254.96
- Task Pane:** Shows a list of tasks, including "Task 1" and "Task 2", with their respective completion status and dates.

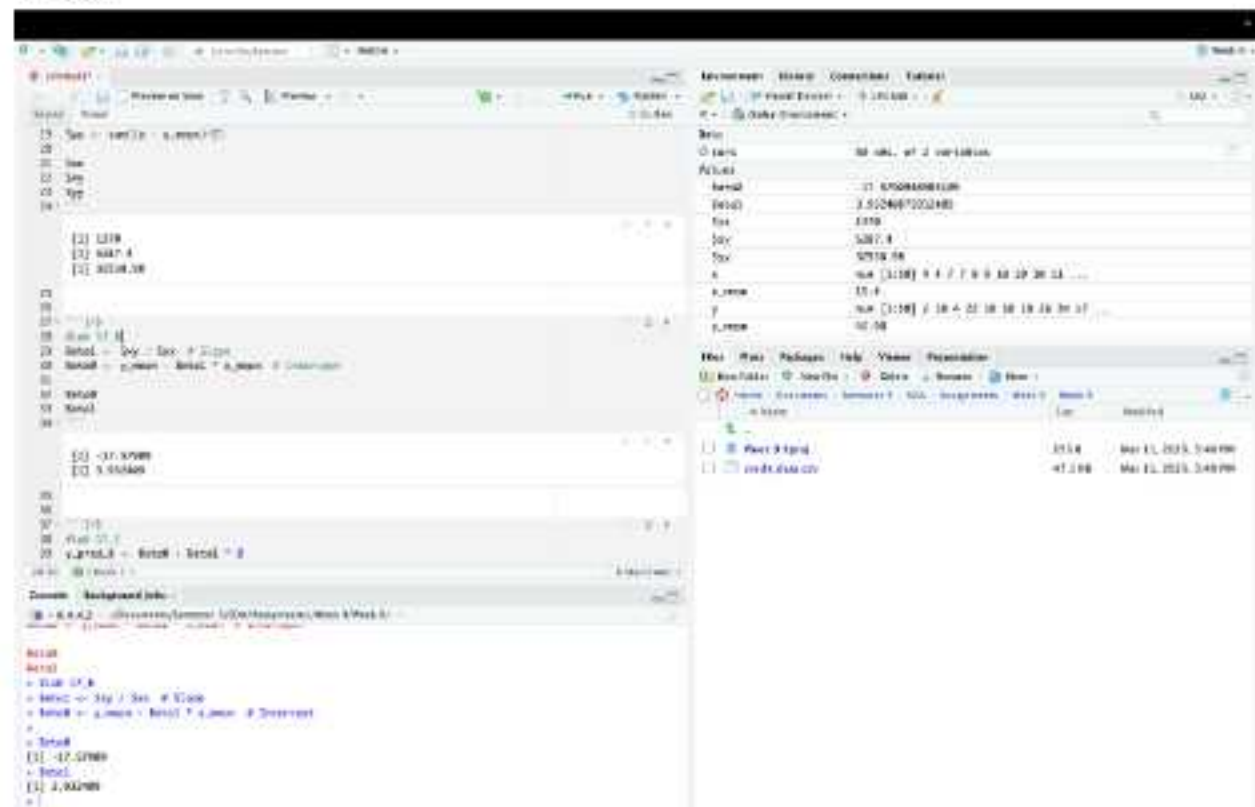
b) Compare these coefficients you obtained via manual calculate to the ones obtained using the `lm()` function used in class. Are they same/similar?

Code:

```
26
27 - ```{r}
28 #Lab 17_8
29 Beta1 <- Sxy / Sxx # Slope
30 Beta0 <- y_mean - Beta1 * x_mean # Intercept
31
32 Beta0
33 Beta1
34 - ```
```

```
[1] -17.57909
```

Output:



Ans: - Intercept (Beta0): -17.57909 and Slope (Beta1): 3.932409

- They are different

c) Predict the stopping distance for a speed of 8 mph. You MUST use the Beta0 and Beta1 you found in this calculation.

Code:

```
36
37 = ```{r}
38 #Lab 17_C
39 y_pred_8 <- Beta0 + Beta1 * 8
40 y_pred_8
41 = ```
```

Output:

The screenshot shows the RStudio environment with the following components:

- Source Editor:** Contains R code for calculating the predicted stopping distance at 8 mph. The code includes comments and variable assignments for Beta0 and Beta1.
- Environment:** Displays the current environment with variables:

Variable	Value
Beta0	-17.57989
Beta1	3.52469
y_pred_8	11.88024
- Console:** Shows the execution of the code, including the calculation of y_pred_8.
- Files:** Lists the files in the project, including 'Part 5.rmd' and '17Lab17_C.R'.

The screenshot shows the RStudio interface with the following content:

Console:

```

> model <- lm(dist ~ speed, data = cars)
> summary(model)

Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min:  -3.25   1st Qu:  -2.12   Median:  0.21   3rd Qu:  1.54   Max:  4.30

Coefficients:
(Intercept) 17.572    0.754    0.000    0.000
speed       0.006    0.001    0.000    0.000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 ''

Residual standard error: 15.34 on 44 degrees of freedom
Multiple R-squared:  0.651,    Adjusted R-squared:  0.640
F-statistic: 85.37 on 1 and 44 DF, p-value: 1.46e-12

```

R Script:

```

# Load the data
cars <- read.csv("cars.csv")

# Fit a linear model
model <- lm(dist ~ speed, data = cars)

# Summary of the model
summary(model)

```

Environment:

Object	Class	Attributes	Value	Environment
cars	data.frame		100 x 2	Global
model	lm		lm object	Global

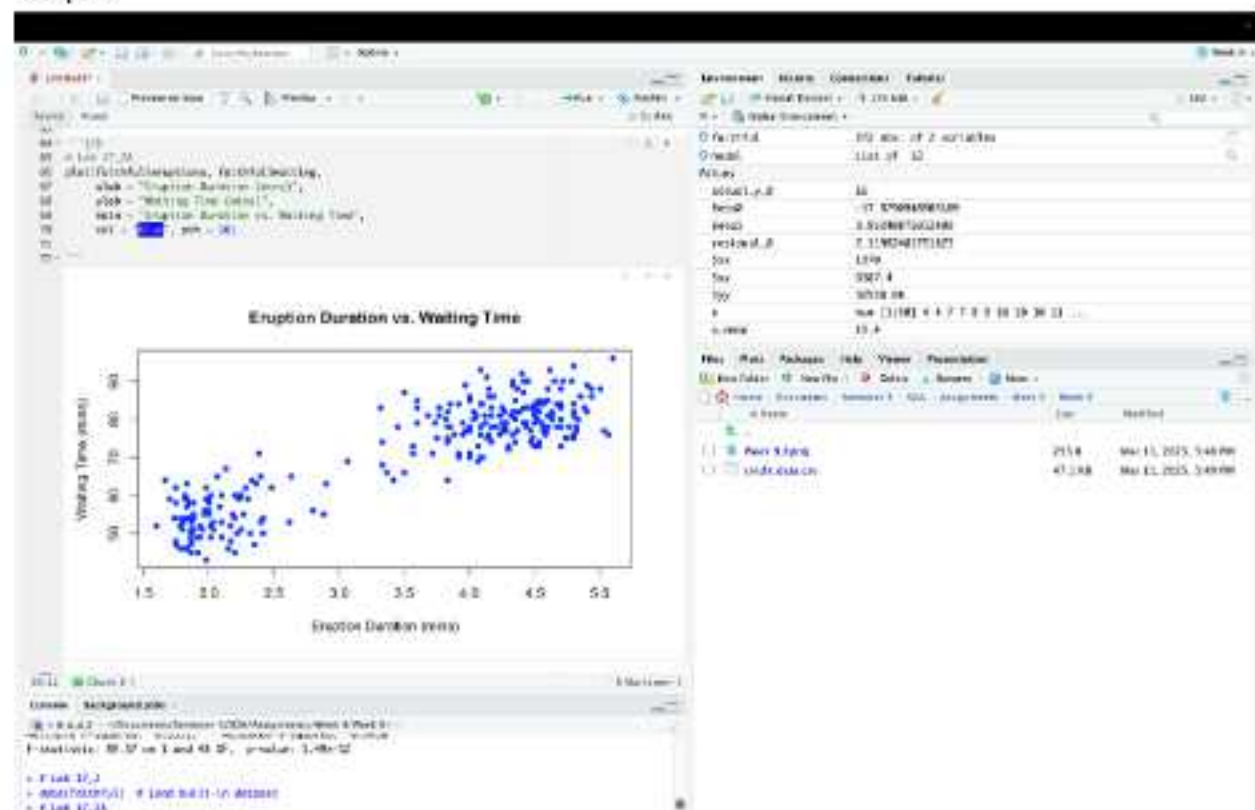
2. Load the faithful dataset by typing `data(faithful)`. Do the following:

a) Plot the waiting times against the duration of eruption.

Code:

```
63
64 = ```{r}
65 # Lab 17_2A
66 plot(faithful$eruptions, faithful$waiting,
67       xlab = "Eruption Duration (mins)",
68       ylab = "Waiting Time (mins)",
69       main = "Eruption Duration vs. Waiting Time",
70       col = "blue", pch = 16)
71
72 = ```
```

Output:

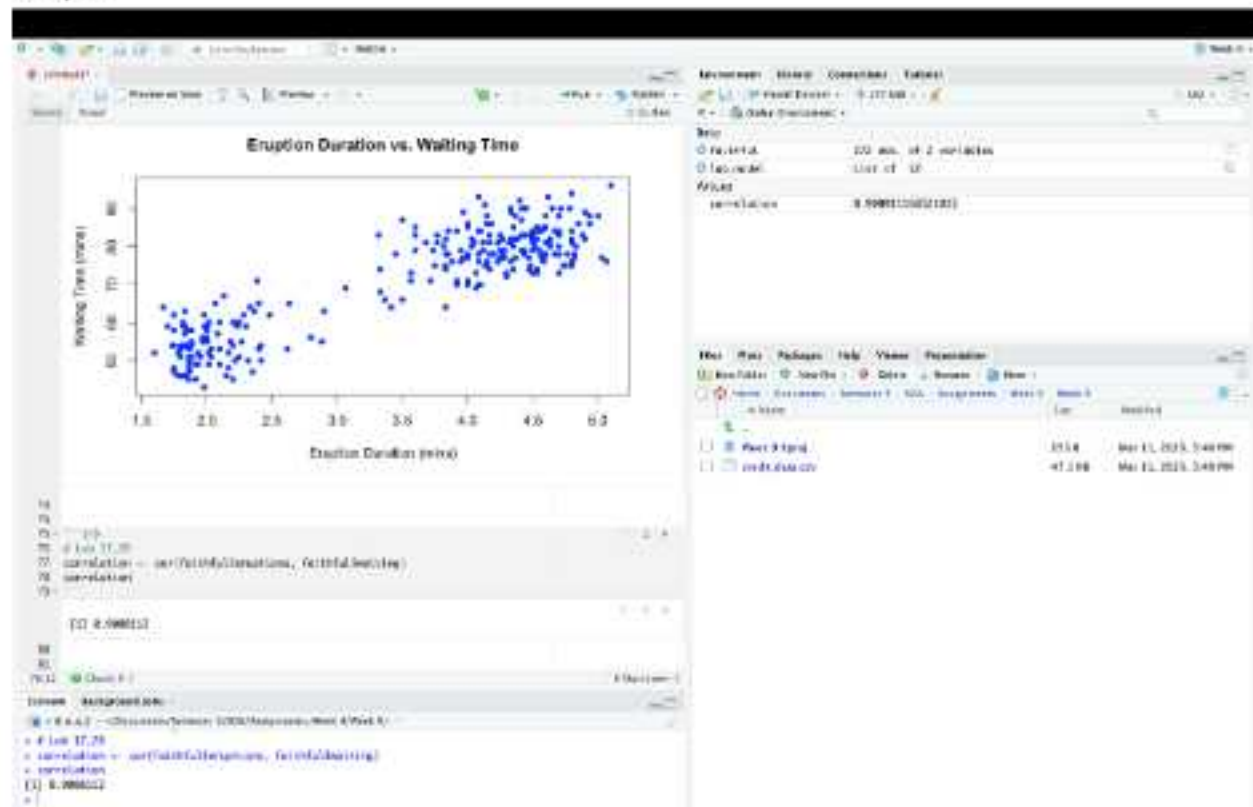


b) Compute the Pearson correlation coefficient between waiting times and duration of eruption. Is there evidence of a linear relationship? Is it: weak, strong, positive, etc.

Code:

```
74
75 = ``{r}
76 # Lab 17_28
77 correlation <- cor(faithful$eruptions, faithful$waiting)
78 correlation
79 = ``
```

Output:



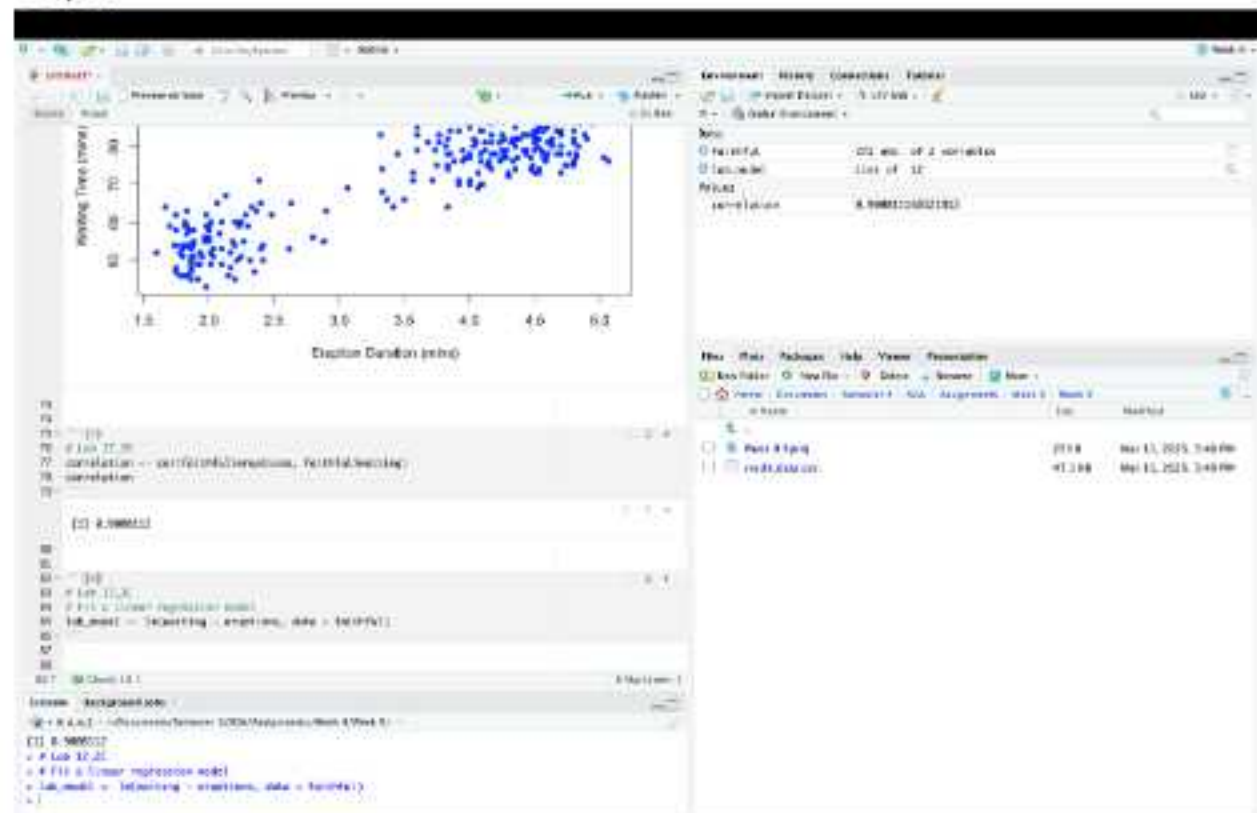
Ans:

c) Use `lm` to fit waiting times as a linear function of eruption durations and save the result of the regression function to the variable 'lab_model'.

Code:

```
80
81
82 > ```{r}
83 # Lab 17_2C
84 # Fit a linear regression model
85 lab_model <- lm(waiting ~ eruptions, data = faithful)
86 > ```
```

Output:



e) Using the 'lab_model' to predict the waiting time for an eruption duration of 3.667 mins.

Code:

```
93
94 = ```{r}
95 # Lab 17_2E
96 new_data <- data.frame(eruptions = 3.667)
97 predicted_waiting <- predict(lab_model, new_data)
98 predicted_waiting
99 = ```
```

Output:

The screenshot shows an RStudio interface with the following components:

- Source Editor:** Contains R code for fitting a linear model and making a prediction.


```
## Fit a linear model
fit1 <- lm(waiting ~ eruptions, data = hawaii)

## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.6796  -4.4831   0.2122   3.8946  15.9779

## Coefficients:
## (Intercept)  eruptions
##          28.1049          2.3698
##          18.7226          0.2048          34.28
##          1.9e-16
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 3.838 on 276 degrees of freedom
## Multiple R-squared:  0.8015, Adjusted R-squared:  0.8000
## F-statistic: 2287 on 1 and 276 DF, p-value: < 2.2e-16
```
- Environment:** Shows the objects created in the environment.

Object	Value
fit1	lm object of class "lm"
new_data	data frame of 1 row
predicted_waiting	numeric vector of 1 element
- Console:** Shows the output of the R code.


```
## Fit a linear model
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.6796  -4.4831   0.2122   3.8946  15.9779

## Coefficients:
## (Intercept)  eruptions
##          28.1049          2.3698
##          18.7226          0.2048          34.28
##          1.9e-16
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 3.838 on 276 degrees of freedom
## Multiple R-squared:  0.8015, Adjusted R-squared:  0.8000
## F-statistic: 2287 on 1 and 276 DF, p-value: < 2.2e-16

## Predicted waiting time for an eruption duration of 3.667 mins
## [1] 22.82098
```
- Files:** Shows the files in the project directory.

3. kNN Preliminaries

a) Write a function that returns the Euclidean distance between two Vectors of the same length. Function must return an error if vectors are of different lengths.

Code:

```

101
102 * '''{r}
103 # Lab 17_3A
104 * euclidean_distance <- function(v1, v2) {
105   # Check if vectors are of the same length
106   * if (length(v1) != length(v2)) {
107     stop("Error: Vectors must be of the same length.")
108   }
109
110   # Compute Euclidean distance
111   distance <- sqrt(sum((v1 - v2)^2))
112   return(distance)
113 * }
114 * '''
115

```

Output:

The screenshot shows the RStudio IDE interface. The left pane displays the R script with the following code:

```

101
102 * '''{r}
103 # Lab 17_3A
104 * euclidean_distance <- function(v1, v2) {
105   # Check if vectors are of the same length
106   * if (length(v1) != length(v2)) {
107     stop("Error: Vectors must be of the same length.")
108   }
109
110   # Compute Euclidean distance
111   distance <- sqrt(sum((v1 - v2)^2))
112   return(distance)
113 * }
114 * '''
115

```

The right pane shows the 'Files' tab, displaying a list of files in the project:

File	Size	Modified
01_data	101 B	Mar 14, 2025, 9:05 PM
02_data	101 B	Mar 14, 2025, 9:05 PM
03_data	101 B	Mar 14, 2025, 9:05 PM
04_data	101 B	Mar 14, 2025, 9:05 PM
05_data	101 B	Mar 14, 2025, 9:05 PM
06_data	101 B	Mar 14, 2025, 9:05 PM
07_data	101 B	Mar 14, 2025, 9:05 PM
08_data	101 B	Mar 14, 2025, 9:05 PM
09_data	101 B	Mar 14, 2025, 9:05 PM
10_data	101 B	Mar 14, 2025, 9:05 PM
11_data	101 B	Mar 14, 2025, 9:05 PM
12_data	101 B	Mar 14, 2025, 9:05 PM
13_data	101 B	Mar 14, 2025, 9:05 PM
14_data	101 B	Mar 14, 2025, 9:05 PM
15_data	101 B	Mar 14, 2025, 9:05 PM
16_data	101 B	Mar 14, 2025, 9:05 PM
17_data	101 B	Mar 14, 2025, 9:05 PM
18_data	101 B	Mar 14, 2025, 9:05 PM
19_data	101 B	Mar 14, 2025, 9:05 PM
20_data	101 B	Mar 14, 2025, 9:05 PM

a) Browse the data, view its structure and its dimensionality. How many variables does it have? How many rows does it have?

Code:

```
137
138 - ```{r}
139 # Lab 17_4A
140 # View structure and dimensionality
141 str(vehicle_data)
142 dim(vehicle_data)
143 - ```
```

Output:

The screenshot shows the RStudio interface. The top pane displays a preview of the 'vehicle_data' data frame with columns: vehicle, price, mileage, engine, horsepower, cylinders, weight, length, and horsepower. The bottom-left pane shows the output of 'str(vehicle_data)' and 'dim(vehicle_data)', indicating 74 observations and 10 variables. The bottom-right pane shows the R console output.

```
data.frame: 74 obs. of  10 variables:
 $ vehicle: chr "Ford" "Ford" "Ford" "Ford" ...
 $ price:   num 4100 4100 4100 4100 ...
 $ mileage:   num  0  0  0  0 ...
 $ engine:    chr  4  4  4  4 ...
 $ horsepower: num  6  6  6  6 ...
 $ cylinders:  num  4  4  4  4 ...
 $ weight:     num 1310 1310 1310 1310 ...
 $ length:     num 400 400 400 400 ...
 $ horsepower: num  52  52  52  52 ...
 $ gear_ratio:  num  2.35 2.35 2.35 2.35 ...
```

```
dim(vehicle_data)
[1] 74 10
```

Ans: It has 74 obs of 10 variables

c) Normalize the data (excluding the "label" variable). The resulting values of all variables should be in the range from 0 to 1.

Code:

```

145
146 = ""{r}
147 #lab 17.4Q
148 # Function to normalize values between 0 and 1
149 normalize <- function(x) {
150   return((x - min(x)) / (max(x) - min(x)))
151 }
152
153 # Create a copy of the dataframe for normalization
154 vehicle_data_norm <- vehicle_data
155
156 # Normalize all numeric columns (exclude the "origin" column)
157 vehicle_data_norm[, 2:ncol(vehicle_data)] <- lapply(vehicle_data[, 2:ncol(vehicle_data)], normalize)
158
159 # Check the normalized data
160 head(vehicle_data_norm)
161 =

```

Output:

The screenshot displays a Jupyter Notebook with the following content:

```

114
115
116 """
117 Plot 2D
118 Plot function for normalizing values between 0 and 1
119 normalise = lambda(x):
120     return(x - min(x)) / (max(x) - min(x))
121 """
122
123 # Create a copy of the dataframe for normalisation
124 vehicle_data_norm = vehicle_data
125
126 # Normalize all numeric columns. Exclude the "color" column
127 vehicle_data_norm[2:10] = vehicle_data_norm[2:10].apply(normalise)
128
129 # Check the normalized data
130 load_data('data.csv')
131 """
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
10
```


e) Build a `knn()` classifier using the "class" package. Set `k=9`.

Code:

```
188
189 = ""{r}
190 # Lab 17_4E
191 # Load required package
192 library(class)
193
194 # Prepare training and test sets
195 train_labels <- train_data$origin
196 test_labels <- test_data$origin
197
198 # Remove the label column from the feature sets
199 train_features <- train_data[, -1] # Excluding the "origin" column
200 test_features <- test_data[, -1]  # Excluding the "origin" column
201
202 # Build KNN model with k=9
203 knn_pred <- knn(train = train_features,
204                 test = test_features,
205                 cl = train_labels,
206                 k = 9)
207 = ""
208
```

Output:

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R code from the previous block, with line numbers 188 to 208. The code loads the 'class' package, prepares training and test sets by removing the 'origin' column, and builds a KNN model with `k=9`.
- Environment:** Displays the objects created in the global environment:
 - `test_data`: 15 obs. of 10 variables
 - `test_features`: 14 obs. of 9 variables
 - `train_data`: 30 obs. of 10 variables
 - `knn_model`: 30 obs. of 9 variables
 - `train_labels`: 1x1 (30, 2) 1 2 4 0 0 7 8 8 10 10 ...
 - `test_labels`: 14 obs. of 10 variables
 - `knn_pred`: 14 obs. of 10 variables
- Console:** Shows the execution output, including package loading messages and the final assignment of `knn_pred`.
- Files:** Lists the files in the project, including `test_data.csv` and `train_data.csv`.

g) What is the accuracy? Is it a good performance?

```

usa      1      8

Accuracy : 0.7857
95% CI : (0.492, 0.9534)
No Information Rate : 0.7143
P-Value [Acc > NIR] : 0.4001

Kappa : 0.5116

Mcnemar's Test P-Value : 1.0000

Sensitivity : 0.7500
Specificity : 0.8000
Pos Pred Value : 0.6000
Neg Pred Value : 0.8889
Prevalence : 0.2857
Detection Rate : 0.2143
Detection Prevalence : 0.3571
Balanced Accuracy : 0.7750

'Positive' Class : other

```

Save CI

Ans: - The accuracy of the model is 0.7857 (78.57%)

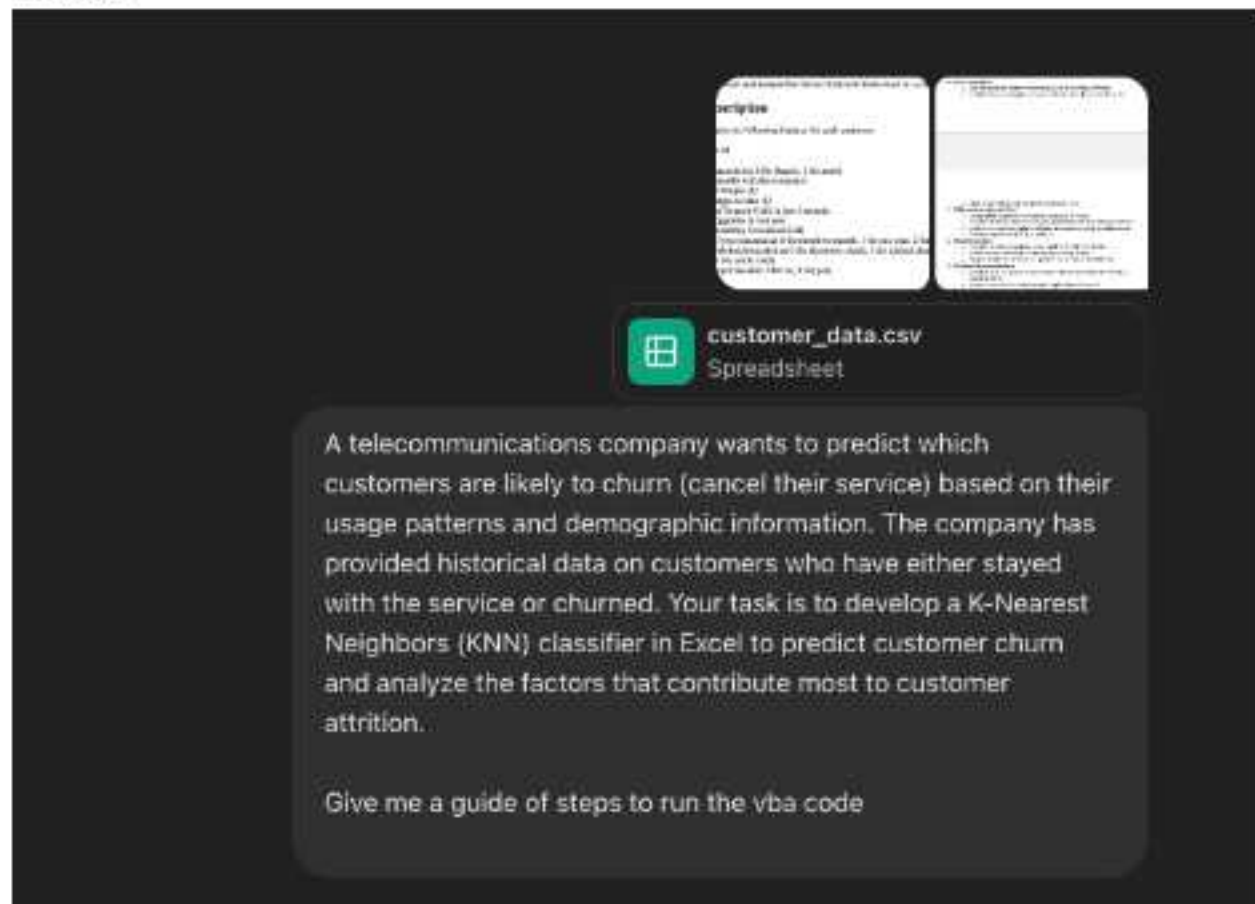
- The model performs moderately well but may not be significantly better than random guessing.
- If the application requires high precision or recall, further model tuning or feature engineering might be needed.

5. Gen AI Prompt Engineering:

A telecommunications company wants to predict which customers are likely to churn (cancel their service) based on their usage patterns and demographic information. The company has provided historical data on customers who have either stayed with the service or churned. Your task is to develop a K-Nearest Neighbors (KNN) classifier in Excel to predict customer churn and analyze the factors that contribute most to customer attrition.

Bot Used: ChatGPT

Prompt:



Bot:

Here's a step-by-step guide on writing VBA code for implementing a nearest neighbors (KNN) classifier in Excel:

Step 1: Enable Developer Tab

1. Open Excel.
2. Click on **File** → **Options**.
3. Go to **Customize Ribbon**.
4. Check **Developer** in the right panel and click **OK**.

Step 2: Open VBA Editor

1. Press **Alt + F11** to open the VBA editor.
2. Click **Insert** → **Module** to create a new module.

Step 3: Load Data into Excel

1. Open your dataset file (e.g., **dataset.csv**) in Excel.
2. Ensure that data is structured properly with column headers matching the dataset description.

Step 4: Write VBA Code for KNN

1. Copy and paste the VBA code for KNN into the module.

Get working:

Search Query

2. The code should:
 - Read the dataset.
 - Normalize numerical features using Min-Max scaling.
 - Compute Euclidean distances between test instances and all training instances.
 - Select the **K** nearest neighbors.
 - Perform majority voting to classify new data.

Step 5: Run the VBA Code

1. Open the VBA editor and return to Excel.
2. Press **Alt + F8**, select the macro you created (e.g., **RunKNN**), and click **Run**.

Step 6: Evaluate Model Performance

1. Compute accuracy, precision, recall, and F1-score.
2. Create a confusion matrix.
3. Analyze feature importance.

Step 7: Generate Business Recommendations

- Use the model insights to provide actionable steps to reduce customer churn.

Would you like me to generate the VBA code for you?

Get working:

Search Query

ChaosPT -

Accepted generating
Here's a VBA implementation of the k-Nearest Neighbors (KNN) classifier in Excel.

Steps Covered in the VBA Code

- Load data from the Excel sheet
- Normalize numerical features using Min-Max Scaling
- Calculate Euclidean distance for each test instance
- Select k-nearest neighbors and classify based on majority voting
- Output predictions in a new column

VBA Code for KNN in Excel

- Open Excel and press ALT + F11 to open the VBA editor.
- Click Insert > Module and paste the following code:

```

Option Explicit

Function KNNClassifier(ByVal value As Double, ByVal actual As Double, ByVal result As Double) As Double
    ' Returns function error/exception
    If result = actual Then
        KNNClassifier = 0
    Else
        KNNClassifier = 1
    End If
End Function

Function EuclideanDistance(ByVal val As Range, Ref1 As Range, Ref2 As Range) As Double
    ' Calculate Euclidean Distance between two rows
    Dim i As Integer
    Dim sum As Double
    sum = 0
    For i = 1 To val.Cells.Count
        sum = sum + (val.Cells(i, 1) - Ref1.Cells(i, 1)) ^ 2 + (val.Cells(i, 2) - Ref2.Cells(i, 2)) ^ 2
    Next i
    EuclideanDistance = Sqr(sum)
End Function
  
```

Upgrade plan
Manage your account

ChaosPT -

Option Explicit

```

Function KNNClassifier(ByVal value As Double, ByVal actual As Double, ByVal result As Double) As Double
    ' Returns function error/exception
    If result = actual Then
        KNNClassifier = 0
    Else
        KNNClassifier = 1
    End If
End Function

Function EuclideanDistance(ByVal val As Range, Ref1 As Range, Ref2 As Range) As Double
    ' Calculate Euclidean Distance between two rows
    Dim i As Integer
    Dim sum As Double
    sum = 0
    For i = 1 To val.Cells.Count
        sum = sum + (val.Cells(i, 1) - Ref1.Cells(i, 1)) ^ 2 + (val.Cells(i, 2) - Ref2.Cells(i, 2)) ^ 2
    Next i
    EuclideanDistance = Sqr(sum)
End Function
  
```

Upgrade plan
Manage your account

Macro to Normalize Features (Min-Max Scaling)

This macro will normalize all numeric features between 0 and 1.

```

--
Sys Name() as FeatureList()
Sys us as Worksheet
Dim lastRow as Long, lastCol as Long
Dim rowVal as Double, rowVal2 as Double
Dim col as Integer, i as Long

Set us = ThisWorkbook.Worksheet("Sheet1") ' Change sheet name if needed
lastRow = us.Cells(10000, 10000).End(xlToLeft).Row
lastCol = us.Cells(1, 10000).End(xlToRight).Column

' Loop through numerical columns (skip categorical ones)
For col = 2 To lastCol - 1
    rowVal = Application.Min(worksheet.Range(us.Cells(2, col), us.Cells(lastRow, col)))
    rowVal2 = Application.Max(worksheet.Range(us.Cells(2, col), us.Cells(lastRow, col)))

    ' Normalize values
    For i = 2 To lastRow
        us.Cells(i, col).Value = (us.Cells(i, col).Value - rowVal) / (rowVal2 - rowVal) + 1
    Next i
Next col
  
```

Run this macro

Macro Recorder: Macro: Macro1, Step: 1 of 1

Macro to Prepare Data (Split into Training & Testing)

This macro will split the dataset into 80% training and 20% testing.

```

--
Sys Properties()
Sys us as Worksheet
Dim lastRow as Long, lastCol as Long
Dim i as Long

Set us = ThisWorkbook.Worksheet("Sheet1") ' Change sheet name if needed
lastRow = us.Cells(10000, 10000).End(xlToLeft).Row
lastCol = us.Cells(1, 10000).End(xlToRight).Column

' Label data split
us.Cells(1, lastCol + 1).Value = "Dataset Split"

For i = 2 To lastRow
    If i = Application.Round(Rnd * (lastRow - 1 + 1)) Then
        us.Cells(i, lastCol + 1).Value = "Training"
    Else
        us.Cells(i, lastCol + 1).Value = "Testing"
    End If
Next i
  
```

Run this macro

Macro Recorder: Macro: Macro1, Step: 1 of 1

ChatGPT -

Macro for K-Nearest Neighbors (KNN) Classification

This macro will:

- Calculate Euclidean distances between test instances and training data.
- Identify nearest neighbors.
- Use majority voting to predict customer churn.

```

Function EvaluateKNN(test As Range, testRow As Integer As Integer)
    Dim k As Integer
    Dim distances As Range
    Dim sorted As Range
    Dim neighbors As Range
    Dim majority As String
    Dim result As String

    k = 5 ' Number of nearest neighbors to consider
    distances = Range(testRow, testRow + k - 1)
    sorted = SortRange(distances, 1, 2)
    neighbors = sorted.Rows(1, k)
    majority = ""
    result = ""

    For i = 1 To k
        majority = majority + neighbors.Cells(i, 1).Value & ", "
    Next i

    majority = Left(majority, Len(majority) - 2)
    result = majority
End Function

Sub KNN_Classify()
    ' Add your code here

```

Add comments

Search

ChatGPT -

VBA Code for Model Evaluation Macro

```

Sub VBA_Model_Evaluation()
    Dim ws As Worksheet, evalSheet As Worksheet
    Dim testRow As Integer, k As Integer
    Dim results As Range, k As Integer
    Dim TP As Integer, FP As Integer, FN As Integer, TN As Integer
    Dim Accuracy As Double, Precision As Double, Recall As Double, F1_Score As Double
    Dim totalPredictions As Integer, correctPredictions As Integer
    Dim resultRow As Integer

    ' Set data sheet
    Set ws = ThisWorkbook.Sheets("CustomerData") ' Adjust the sheet name

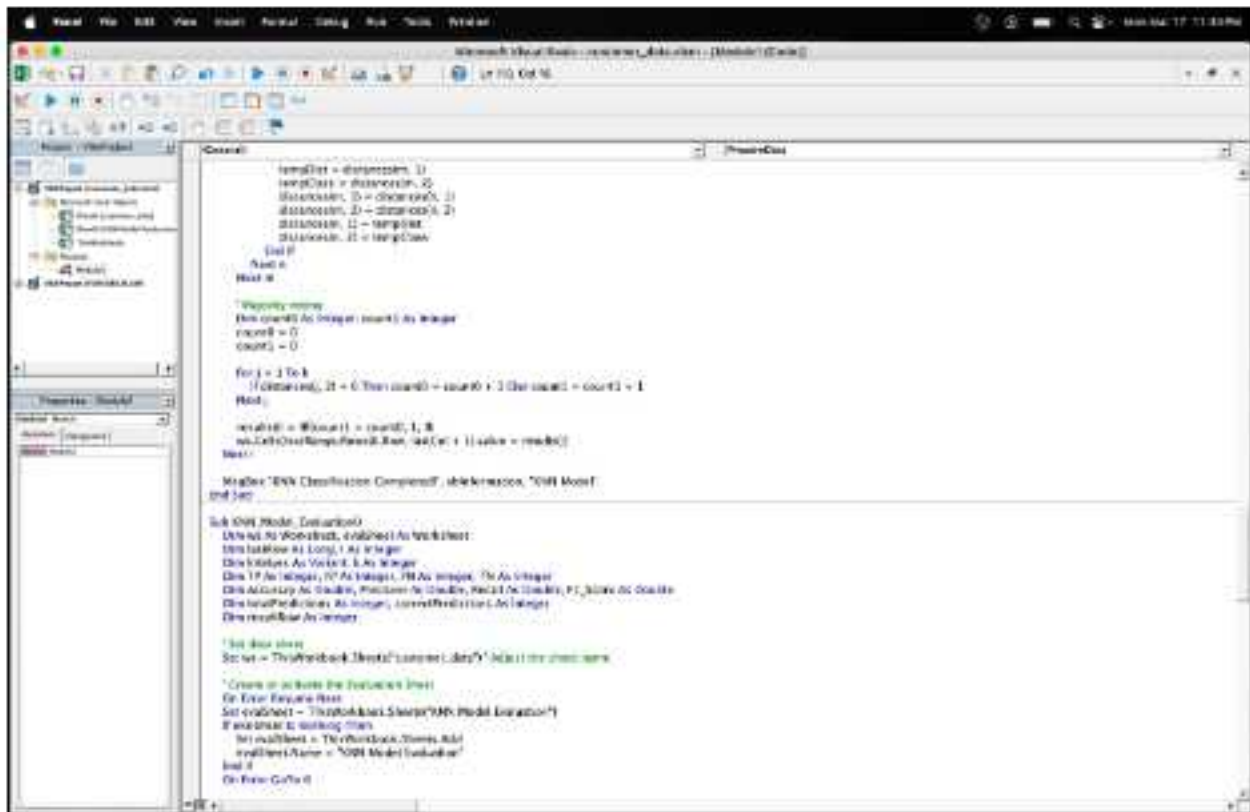
    ' Create an array for Evaluation Sheet
    Dim evalArray As Variant
    Set evalSheet = ThisWorkbook.Sheets("Model Evaluation")
    If evalSheet Is Nothing Then
        Set evalSheet = ThisWorkbook.Sheets.Add
        evalSheet.Name = "Model Evaluation"
    End If
    Dim evalRow As Integer

    ' Clear old results

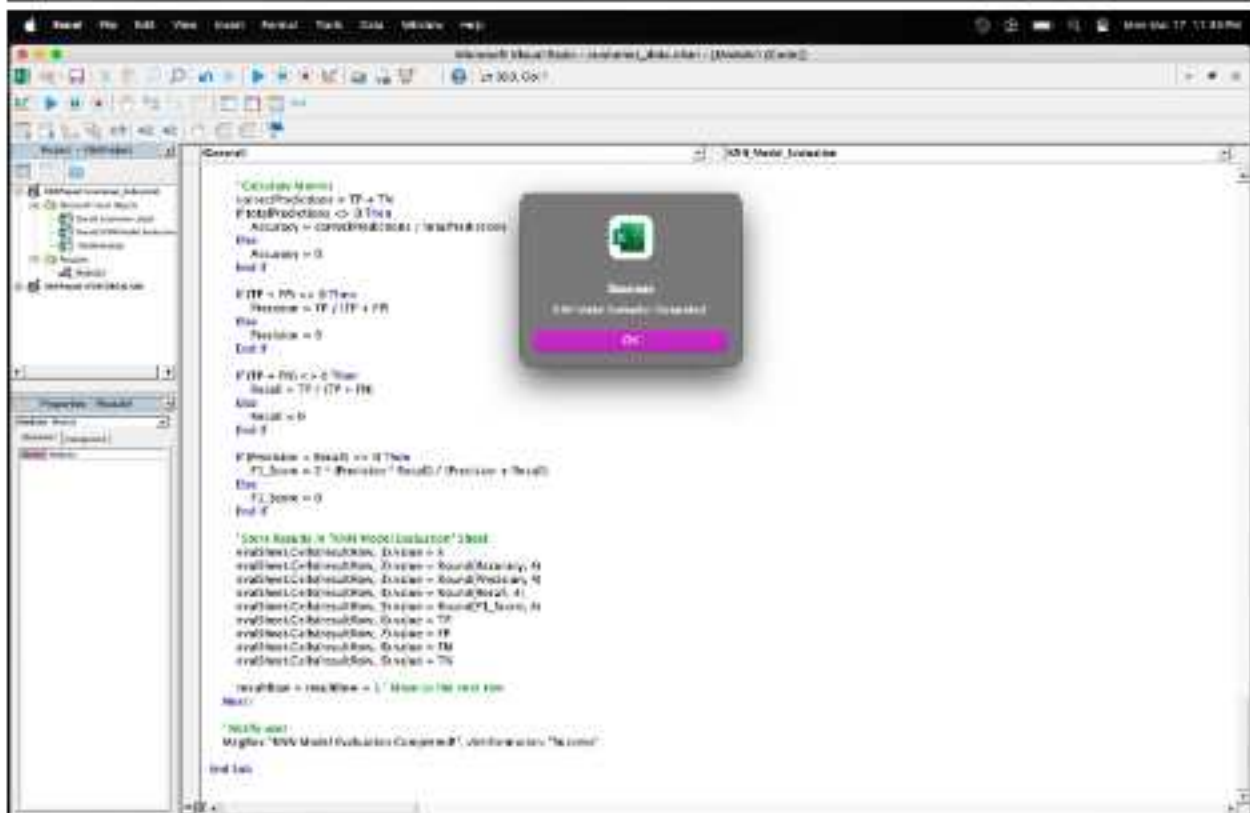
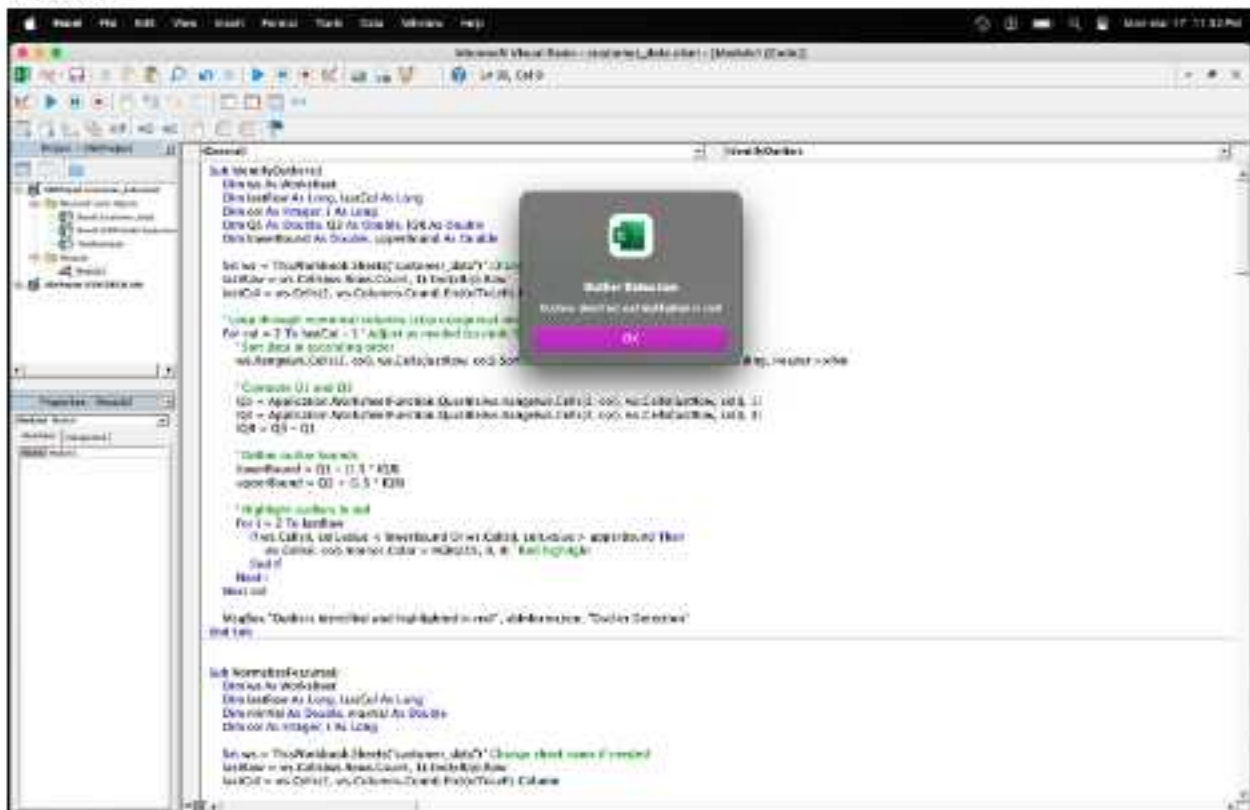
```

Add comments

Search



Msgbox:



Output:

Customer ID	Age	Gender	Name	Monthly Credit Card Payments	Credit Score	Credit Limit	Credit Used	Credit Available	Credit History	Credit Status
1	25	M	John Doe	100	750	1000	250	750	Good	Active
2	30	F	Jane Smith	150	720	1200	300	900	Good	Active
3	35	M	Mike Johnson	200	680	1500	400	1100	Fair	Active
4	40	F	Sarah Brown	250	650	1800	500	1300	Fair	Active
5	45	M	David Wilson	300	620	2000	600	1400	Fair	Active
6	50	F	Linda Davis	350	600	2200	700	1500	Fair	Active
7	55	M	Robert Miller	400	580	2400	800	1600	Fair	Active
8	60	F	Patricia Moore	450	560	2600	900	1700	Fair	Active
9	65	M	Thomas Taylor	500	540	2800	1000	1800	Fair	Active
10	70	F	Barbara White	550	520	3000	1100	1900	Fair	Active
11	75	M	Charles Black	600	500	3200	1200	2000	Fair	Active
12	80	F	Elizabeth Green	650	480	3400	1300	2100	Fair	Active
13	85	M	William King	700	460	3600	1400	2200	Fair	Active
14	90	F	Ann Lee	750	440	3800	1500	2300	Fair	Active
15	95	M	George Hall	800	420	4000	1600	2400	Fair	Active
16	100	F	Karen Young	850	400	4200	1700	2500	Fair	Active
17	105	M	Edward Scott	900	380	4400	1800	2600	Fair	Active
18	110	F	Michelle Adams	950	360	4600	1900	2700	Fair	Active
19	115	M	Christopher Baker	1000	340	4800	2000	2800	Fair	Active
20	120	F	Stephanie Carter	1050	320	5000	2100	2900	Fair	Active
21	125	M	Matthew Evans	1100	300	5200	2200	3000	Fair	Active
22	130	F	Olivia Hill	1150	280	5400	2300	3100	Fair	Active
23	135	M	Andrew Jackson	1200	260	5600	2400	3200	Fair	Active
24	140	F	Sophia King	1250	240	5800	2500	3300	Fair	Active
25	145	M	Benjamin Lee	1300	220	6000	2600	3400	Fair	Active
26	150	F	Evelyn Miller	1350	200	6200	2700	3500	Fair	Active
27	155	M	Samuel Moore	1400	180	6400	2800	3600	Fair	Active
28	160	F	Victoria Taylor	1450	160	6600	2900	3700	Fair	Active
29	165	M	William White	1500	140	6800	3000	3800	Fair	Active
30	170	F	Isabella Black	1550	120	7000	3100	3900	Fair	Active
31	175	M	James Brown	1600	100	7200	3200	4000	Fair	Active
32	180	F	Aria Davis	1650	80	7400	3300	4100	Fair	Active
33	185	M	Lucas Evans	1700	60	7600	3400	4200	Fair	Active
34	190	F	Charlotte Hill	1750	40	7800	3500	4300	Fair	Active
35	195	M	Henry Jackson	1800	20	8000	3600	4400	Fair	Active
36	200	F	Amelia King	1850	0	8200	3700	4500	Fair	Active
37	205	M	Sebastian Lee	1900	0	8400	3800	4600	Fair	Active
38	210	F	Madeline Miller	1950	0	8600	3900	4700	Fair	Active
39	215	M	Leo Moore	2000	0	8800	4000	4800	Fair	Active
40	220	F	Grace Taylor	2050	0	9000	4100	4900	Fair	Active
41	225	M	Isaac White	2100	0	9200	4200	5000	Fair	Active
42	230	F	Chloe Black	2150	0	9400	4300	5100	Fair	Active
43	235	M	Wyatt Brown	2200	0	9600	4400	5200	Fair	Active
44	240	F	Zoe Davis	2250	0	9800	4500	5300	Fair	Active
45	245	M	Julian Evans	2300	0	10000	4600	5400	Fair	Active
46	250	F	Alexis Hill	2350	0	10200	4700	5500	Fair	Active
47	255	M	Samuel Jackson	2400	0	10400	4800	5600	Fair	Active
48	260	F	Madison King	2450	0	10600	4900	5700	Fair	Active
49	265	M	David Lee	2500	0	10800	5000	5800	Fair	Active
50	270	F	Olivia Miller	2550	0	11000	5100	5900	Fair	Active
51	275	M	Christopher Moore	2600	0	11200	5200	6000	Fair	Active
52	280	F	Isabella Taylor	2650	0	11400	5300	6100	Fair	Active
53	285	M	Benjamin White	2700	0	11600	5400	6200	Fair	Active
54	290	F	Evelyn Black	2750	0	11800	5500	6300	Fair	Active
55	295	M	Samuel Brown	2800	0	12000	5600	6400	Fair	Active
56	300	F	Victoria Davis	2850	0	12200	5700	6500	Fair	Active
57	305	M	William Evans	2900	0	12400	5800	6600	Fair	Active
58	310	F	Isabella Hill	2950	0	12600	5900	6700	Fair	Active
59	315	M	James Jackson	3000	0	12800	6000	6800	Fair	Active
60	320	F	Aria King	3050	0	13000	6100	6900	Fair	Active
61	325	M	Lucas Lee	3100	0	13200	6200	7000	Fair	Active
62	330	F	Charlotte Miller	3150	0	13400	6300	7100	Fair	Active
63	335	M	Henry Moore	3200	0	13600	6400	7200	Fair	Active
64	340	F	Amelia Taylor	3250	0	13800	6500	7300	Fair	Active
65	345	M	David White	3300	0	14000	6600	7400	Fair	Active
66	350	F	Olivia Black	3350	0	14200	6700	7500	Fair	Active
67	355	M	Christopher Brown	3400	0	14400	6800	7600	Fair	Active
68	360	F	Isabella Davis	3450	0	14600	6900	7700	Fair	Active
69	365	M	Benjamin Evans	3500	0	14800	7000	7800	Fair	Active
70	370	F	Evelyn Hill	3550	0	15000	7100	7900	Fair	Active
71	375	M	Samuel Jackson	3600	0	15200	7200	8000	Fair	Active
72	380	F	Victoria King	3650	0	15400	7300	8100	Fair	Active
73	385	M	William Lee	3700	0	15600	7400	8200	Fair	Active
74	390	F	Isabella Miller	3750	0	15800	7500	8300	Fair	Active
75	395	M	James Moore	3800	0	16000	7600	8400	Fair	Active
76	400	F	Aria Taylor	3850	0	16200	7700	8500	Fair	Active
77	405	M	Lucas White	3900	0	16400	7800	8600	Fair	Active
78	410	F	Charlotte Black	3950	0	16600	7900	8700	Fair	Active
79	415	M	Henry Brown	4000	0	16800	8000	8800	Fair	Active
80	420	F	Amelia Davis	4050	0	17000	8100	8900	Fair	Active
81	425	M	David Evans	4100	0	17200	8200	9000	Fair	Active
82	430	F	Olivia Hill	4150	0	17400	8300	9100	Fair	Active
83	435	M	Christopher Jackson	4200	0	17600	8400	9200	Fair	Active
84	440	F	Isabella King	4250	0	17800	8500	9300	Fair	Active
85	445	M	Benjamin Lee	4300	0	18000	8600	9400	Fair	Active
86	450	F	Evelyn Miller	4350	0	18200	8700	9500	Fair	Active
87	455	M	Samuel Moore	4400	0	18400	8800	9600	Fair	Active
88	460	F	Victoria Taylor	4450	0	18600	8900	9700	Fair	Active
89	465	M	William White	4500	0	18800	9000	9800	Fair	Active
90	470	F	Isabella Black	4550	0	19000	9100	9900	Fair	Active
91	475	M	James Brown	4600	0	19200	9200	10000	Fair	Active
92	480	F	Aria Davis	4650	0	19400	9300	10100	Fair	Active
93	485	M	Lucas Evans	4700	0	19600	9400	10200	Fair	Active
94	490	F	Charlotte Hill	4750	0	19800	9500	10300	Fair	Active
95	495	M	Henry Jackson	4800	0	20000	9600	10400	Fair	Active
96	500	F	Amelia King	4850	0	20200	9700	10500	Fair	Active
97	505	M	David Lee	4900	0	20400	9800	10600	Fair	Active
98	510	F	Olivia Miller	4950	0	20600	9900	10700	Fair	Active
99	515	M	Christopher Moore	5000	0	20800	10000	10800	Fair	Active
100	520	F	Isabella Taylor	5050	0	21000	10100	10900	Fair	Active

Customer ID	Age	Gender	Name	Monthly Credit Card Payments	Credit Score	Credit Limit	Credit Used	Credit Available	Credit History	Credit Status
1	25	M	John Doe	100	750	1000	250	750	Good	Active
2	30	F	Jane Smith	150	720	1200	300	900	Good	Active
3	35	M	Mike Johnson	200	680	1500	400	1100	Fair	Active
4	40	F	Sarah Brown	250	650	1800	500	1300	Fair	Active
5	45	M	David Wilson	300	620	2000	600	1400	Fair	Active
6	50	F	Linda Davis	350	600	2200	700	1500	Fair	Active
7	55	M	Robert Miller	400	580	2400	800	1600	Fair	Active
8	60	F	Patricia Moore	450	560	2600	900	1700	Fair	Active
9	65	M	Thomas Taylor	500	540	2800	1000	1800	Fair	Active
10	70	F	Barbara White	550	520	3000	1100	1900	Fair	Active
11	75	M	Charles Black	600	500	3200	1200	2000	Fair	Active
12	80	F	Elizabeth Green	650	480	3400	1300	2100	Fair	Active
13	85	M	William King	700	460	3600	1400	2200	Fair	Active
14	90	F	Ann Lee	750	440	3800	1500	2300	Fair	Active
15	95	M	George Hall	800	420	4000	1600	2400	Fair	Active
16	100	F	Karen Young	850	400	4200	1700	2500	Fair	Active
17	105	M	Edward Scott	900	380	4400	1800	2600	Fair	Active
18	110	F	Michelle Adams	950	360	4600	1900	2700	Fair	Active
19	115	M	Christopher Baker	1000	340	4800	2000	2800	Fair	Active
20	120	F	Stephanie Carter	1050	320	5000	2100	2900	Fair	Active
21	125	M	Matthew Evans	1100	300	5200	2200	3000	Fair	Active
22	130	F	Olivia Hill	1150	280	5400	2300	3100	Fair	Active
23	135	M	Andrew Jackson	1200	260	5600	2400	3200	Fair	Active
24	140	F	Sophia King	1250	240	5800	2500	3300	Fair	Active
25	145	M	Benjamin Lee	1300	220	6000	2600	3400	Fair	Active
26	150	F	Evelyn Miller	1350	200	6200	2700	3500	Fair	Active
27	155	M	Samuel Moore	1400	180	6400	2800	3600	Fair	Active
28	160	F	Victoria Taylor	1450	160	6600	2900	3700	Fair	Active
29	165	M	William White	1500	140	6800	3000	3800	Fair	Active
30	170	F	Isabella Black	1550	120	7000	3100	3900	Fair	Active
31	175	M	James Brown	1600	100	7200	3200	4000	Fair	Active
32	180	F	Aria Davis	1650	80	7400	3300	4100	Fair	Active
33	185	M	Lucas Evans	1700	60	7600	3400	4200	Fair	Active
34	190	F	Charlotte Hill	1750	40	7800	3500	4300	Fair	Active
35	195	M	Henry Jackson	1800	20	8000	3600	4400	Fair	Active
36	200	F	Amelia King	1850	0	8200	3700	4500	Fair	Active
37	205	M	Sebastian Lee	1900	0	8400	3800	4600	Fair	Active
38	210	F	Madeline Miller	1950	0	8600	3900	4700	Fair	Active
39	215	M	Christopher King	2000	0	8800	4000	4800	Fair	Active
40	220	F	Isabella Brown	2050	0	9000	4100	4900	Fair	Active
41	225	M	Matthew White	2100	0	9200	4200	5000	Fair	Active
42	230	F	Olivia Black	2150	0	9400	4300	5100	Fair	Active
43	235	M	Andrew Green	2200	0	9600	4400	5200	Fair	Active
44	240	F	Sophia White	2250	0	9800	4500	5300	Fair	Active
45	245	M	Benjamin Black	2300	0	10000	4600	5400	Fair	Active
46	250	F	Evelyn Green	2350	0	10200	4700	5500	Fair	Active
47	255	M	Samuel White	2400	0	10400	4800	5600	Fair	Active
48	260	F	Victoria Black	2450	0	10600	4900	5700	Fair	Active
49	265	M	William Green	2500	0	10800	5000	5800	Fair	Active
50	270	F	Isabella White	2550	0	11000	5100	5900	Fair	Active
51	275	M	James Black	2600	0	11200	5200	6000	Fair	Active
52	280	F	Aria Green	2650	0	11400	5300	6100	Fair	Active
53	285	M	Lucas White	2700	0	11600	5400	6200	Fair	Active
54	290	F	Charlotte Black	2750	0	11800	5500	6300	Fair	Active
55	295	M	Henry Green	2800	0	12000	5600	6400	Fair	Active
56	300	F	Amelia White	2850	0	12200	5700	6500	Fair	Active
57	305	M	Sebastian Black	2900	0	12400	5800	6600	Fair	Active
58	310	F	Madeline Green	2950	0	12600	5900	6700	Fair	Active
59	315	M	Christopher White	3000	0	12800	6000	6800	Fair	Active
60	320	F	Isabella Black	3050	0	13000	6100	6900	Fair	Active
61	325	M	Matthew Green	3100	0	13200	6200	7000	Fair	Active
62	330	F	Olivia White	3150	0	13400	6300	7100	Fair	Active
63	335	M	Andrew Black	3200	0	13600	6400	7200	Fair	Active
64	340	F	Sophia Green	3250	0	13800	6500	7300	Fair	Active
65	345	M	Benjamin White	3300	0	14000	6600	7400	Fair	Active
66	350	F	Evelyn Black	3350	0	14200	6700	7500	Fair	Active
67	355	M	Samuel Green	3400	0	14400	6800	7600	Fair	Active
68	360	F	Victoria White	3450	0	14600	6900	7700	Fair	Active
69	365	M	William Black	3500	0	14800	7000	7800	Fair	Active
70	370	F	Isabella Green	3550	0	15000	7100	7900	Fair	Active
71	375	M	James White	3600	0	15200	7200	8000	Fair	Active
72	380	F	Aria Black	3650	0	15400	7300	8100	Fair	Active
73	385	M	Lucas Green	3700	0	15600	7400	8200	Fair	Active
74	390	F	Charlotte White	3750	0	15800	7500	8300	Fair	Active
75	395	M	Henry Black	3800	0	16000	7600	8400	Fair	Active
76	400	F	Amelia Green	3850	0	16200	7700	8500	Fair	Active
77	405	M	Sebastian White	3900	0	16400	7800	8600	Fair	Active
78	410	F	Madeline Black	3950	0	16600	7900	8700	Fair	Active
79	415	M	Christopher Green	4000	0	16800	8000	8800	Fair	Active
80	420	F	Isabella White	4050	0	17000	8100	8900	Fair	Active
81	425	M	Matthew Black	4100	0	17200	8200	9000	Fair	Active
82	430	F	Olivia Green	4150	0	17400	8300	9100	Fair	Active
83	435	M	Andrew White	4200	0	17600	8400	9200	Fair	Active
84	440	F	Sophia Black	4250	0	17800	8500	9300	Fair	Active
85	445	M	Benjamin Green	4300	0	18000	8600	9400	Fair	Active
86	450	F	Evelyn White	4350	0	18200	8700	9500	Fair	Active
87	455	M	Samuel Black	4400	0	18400	8800	9600	Fair	Active
88	460	F	Victoria Green	4450	0	18600	8900	9700	Fair	Active
89	465	M	William White	4500	0	18800	9000	9800	Fair	Active
90	470	F	Isabella Black	4550	0	19000	9100	9900	Fair	Active
91	475	M	James Green	4600	0	19200	9200	10000	Fair	Active
92	480	F	Aria White	4650	0	19400	9300	10100	Fair	Active
93	485	M	Lucas Black	4700	0	19600	9400	10200	Fair	Active
94	490	F	Charlotte Green	4750	0	19800	9500	10300	Fair	Active
95	495	M	Henry White	4800	0	20000	9600	10400	Fair	Active
96	500	F	Amelia Black	4850	0	20200	9700	10500	Fair	Active
97	505	M	Sebastian Green	4900	0	20400	9800	10600	Fair	Active
98	510	F	Madeline White	4950	0	20600	9900	10700	Fair	Active
99	515	M	Christopher Black	5000	0	20800	10000	10800	Fair	Active
100	520	F	Isabella Green	5050	0	21000	10100	10900	Fair	Active

6. **Gen AI Prompt Engineering:** use logistic regression to predict the churn probability (between 0 and 1)

Bot Used: Claude.ai

Prompt:

customer_data

CSV

S A telecommunications company wants to predict which customers are likely to churn (cancel their service) based on their usage patterns and demographic information. The company has provided historical data on customers who have either stayed with the service or churned. Your task is to develop a K-Nearest Neighbors (KNN) classifier in Excel to predict customer churn and analyze the factors that contribute most to customer attrition. But you have to Use logistic regression to predict the churn probability (between 0 and 1)

Edit

Claude Profiting Telecom Customer Churn with Logistic Regression Share ☆ 🔍

Logistic Regression Implementation in Excel

Here's how to implement logistic regression in Excel:

- Data Preparation**
 - Import the CSV data into Excel.
 - Split into training and testing sets (80/20 split).
- Feature Selection & Scaling**
 - Exclude Identifier/ID from predictive features.
 - Consider scaling numerical features (Age, Tenure, Monthly_Charge, etc.).
- Model Building using Excel's Solver**

[Excel Logistic Regression Implementation Steps](#)

Feature Importance Analysis

After building the model, you can analyze which factors contribute most to churn:

- Coefficient Analysis:** The magnitude of coefficients indicates feature importance. Larger absolute values suggest stronger influence.
- Odds Ratio Calculation:** Calculating $e^{\text{coefficient}}$ provides odds ratios, which tell you how much the odds of churning change when a feature increases by one unit.

Key Insights from Data Patterns

Ready to Chat...

Claude AI Search 🔍 📄 📊 📝

VM Logistic Regression for Churn Prediction

```

// Importing libraries
library(gamlss)
library(MASS)
library(MASS)

# Loading data
data <- read.csv("churn_data.csv")

# Feature Selection
features <- c("Age", "Tenure", "Monthly_Charge", "Number_of_Reports")
data <- data[,features]

# Scaling features
scale_data <- scale(data)

# Fitting the Logistic Regression Model
logit_model <- glm(scale_data[,1] ~ scale_data[,2:5], family = "binomial")

# Predicting Churn
predicted_churn <- predict(logit_model, newdata = scale_data[,2:5])

# Converting predicted values to 0 or 1
predicted_churn <- ifelse(predicted_churn > 0, 1, 0)

# Calculating Accuracy
accuracy <- sum(predicted_churn == data[,1]) / nrow(data)

# Calculating AUC
auc <- roc_auc_ordinal(predicted_churn, data[,1])

# Calculating Odds Ratios
odds_ratios <- exp(coef(logit_model))

# Calculating P-values
p_values <- pvalues(logit_model)

# Summary
cat("Accuracy:", accuracy, "\n")
cat("AUC:", auc, "\n")
cat("Odds Ratios:", odds_ratios, "\n")
cat("P-values:", p_values, "\n")

```

Claude Profiting Telecom Customer Churn with Logistic Regression Share ☆ 🔍

Feature Importance Analysis

After building the model, you can analyze which factors contribute most to churn:

- Coefficient Analysis:** The magnitude of coefficients indicates feature importance. Larger absolute values suggest stronger influence.
- Odds Ratio Calculation:** Calculating $e^{\text{coefficient}}$ provides odds ratios, which tell you how much the odds of churning change when a feature increases by one unit.

Key Insights from Data Patterns

Based on the data, I've already identified some patterns that likely contribute to churn:

- Contract Type Impact:** Customers with shorter contracts (Contract_Type=1) appear more likely to churn than those with longer contracts.
- Month-Tenure Vulnerability:** New customers (low tenure) are more likely to churn.
- Support Call Correlation:** Customers with more support calls (Support_Calls) have a higher proportion of churn.
- Service Upgrade Relationship:** Customers with intent to upgrade (Service_Upgrade) appear more likely to churn.

Recommended Excel Implementation

Ready to Chat...

Claude AI Search 🔍 📄 📊 📝

VM Logistic Regression for Churn Prediction

```

// Calculating Accuracy
accuracy <- sum(predicted_churn == data[,1]) / nrow(data)

// Calculating AUC
auc <- roc_auc_ordinal(predicted_churn, data[,1])

// Calculating Odds Ratios
odds_ratios <- exp(coef(logit_model))

// Calculating P-values
p_values <- pvalues(logit_model)

// Summary
cat("Accuracy:", accuracy, "\n")
cat("AUC:", auc, "\n")
cat("Odds Ratios:", odds_ratios, "\n")
cat("P-values:", p_values, "\n")

```



```

Microsoft Visual Studio - Microsoft.VisualBasic - [Book1.xl] (Global)
Ln 1, Col 1

'Accuracy = (TP + TN) / (TP + TN + FP + FN)
If TP + FN > 0 Then precision = TP / (TP + FN) Else precision = 0
If TP + TN > 0 Then recall = TP / (TP + FP) Else recall = 0
If precision > 0 And recall > 0 Then
    F1Score = 2 * (precision * recall) / (precision + recall)
Else
    F1Score = 0
End If

'Create Confusion Matrix
On Error Resume Next
Set actual = Worksheets("Data").Range("A2:A1000")
If actual Is Nothing Then
    Set actual = ThisWorkbook.Workbooks.Add
    actual.Name = "Logistic Regression Model"
End If
On Error Goto 0

'Work with data
actual.Clear
actual.Range("A2").Value = "Status"
actual.Range("B2").Value = "Actual"
actual.Range("A3").Value = "Accuracy"
actual.Range("B3").Value = "Precision"
actual.Range("A4").Value = "Recall"
actual.Range("B4").Value = "F1Score"
actual.Range("A5").Value = "ROC"
actual.Range("B5").Value = "AUC"
actual.Range("A6").Value = "Brier Score"
actual.Range("B6").Value = "Confusion Matrix"
actual.Range("A7").Value = "True Positives"
actual.Range("B7").Value = "True Negatives"
actual.Range("A8").Value = "False Positives"
actual.Range("B8").Value = "False Negatives"
actual.Range("A9").Value = "TP"
actual.Range("B9").Value = "TN"
actual.Range("A10").Value = "FP"
actual.Range("B10").Value = "FN"
actual.Range("C2").Value = "TP"

Logistics "Logistic Regression Model Evaluation Complete!", actual.Range("A2")

End Sub

```

Output:

The screenshot shows a Microsoft Excel window with a blank worksheet. The following formulas are visible in the top left corner:

Cell	Formula
A1	=SUM(A2:A10)
B1	=AVERAGE(B2:B10)
C1	=MAX(C2:C10)
D1	=MIN(D2:D10)
E1	=COUNT(E2:E10)
F1	=COUNTIF(F2:F10, >5)
G1	=COUNTIF(G2:G10, <10)
H1	=COUNTIF(H2:H10, <=10)
I1	=COUNTIF(I2:I10, >=10)
J1	=COUNTIF(J2:J10, <=10)
K1	=COUNTIF(K2:K10, >=10)
L1	=COUNTIF(L2:L10, <=10)
M1	=COUNTIF(M2:M10, >=10)
N1	=COUNTIF(N2:N10, <=10)
O1	=COUNTIF(O2:O10, >=10)
P1	=COUNTIF(P2:P10, <=10)
Q1	=COUNTIF(Q2:Q10, >=10)
R1	=COUNTIF(R2:R10, <=10)
S1	=COUNTIF(S2:S10, >=10)
T1	=COUNTIF(T2:T10, <=10)
U1	=COUNTIF(U2:U10, >=10)
V1	=COUNTIF(V2:V10, <=10)
W1	=COUNTIF(W2:W10, >=10)
X1	=COUNTIF(X2:X10, <=10)
Y1	=COUNTIF(Y2:Y10, >=10)
Z1	=COUNTIF(Z2:Z10, <=10)

FileEditViewInsertFormatDataToolsWindowHelp

Microsoft Excel - [Book1.xlsx] - 100%

FileEditViewInsertFormatDataToolsWindowHelp

HomeInsertLayoutReferencesSendToBackFormat CellsFontBackground StylesCells

Font Background Styles Cells

Font Background Styles Cells

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

Formulas

<

The screenshot displays a Windows 10 desktop environment. The active application is Microsoft Excel, showing a spreadsheet titled 'Project_2019_10_10.xlsx'. The spreadsheet has 10 columns and 20 rows of data. The data is organized into a table with various numerical values, some of which are highlighted in yellow. The Excel interface includes the ribbon (File, Home, Insert, Layout, References, Formulas, Data, Review, View, Developer) and the status bar at the bottom. The file name is 'Project_2019_10_10.xlsx'.

Lab 18

Task 1:

Use the 'countries_data.xlsx' dataset.

Code:

```

5
6- ```{r}
7 # Load required libraries
8 library(readxl)
9 library(dplyr)
10- ```
11
12
13- ```{r}
14 # Lab 18 Task 1
15 file_path <- "countries_data.xlsx" # Update with the correct path if needed
16 df <- read_excel(file_path)
17- ```
18

```

Output:

The screenshot shows an RStudio interface with the following components:

- Source Editor:** Contains the R code from the task, with line numbers 5 through 18. The code loads the 'readxl' and 'dplyr' libraries and reads the 'countries_data.xlsx' file into a data frame 'df'.
- Console:** Shows the execution of the code, with the final line being 'df'.
- Environment Pane:** Displays the objects in the global environment. It shows 'df' as a data frame with 24 variables. Below this, a list of variables is shown with their data types and sizes.

Variable	Class	Size
id	integer	1000
iso_alpha2	character	1000
iso_alpha3	character	1000
iso_numeric	integer	1000
lat	double	1000
lon	double	1000
name	character	1000
continent	character	1000
area_sqkm	double	1000
population	double	1000
gdp_millions_usd	double	1000
gdp_per_capita_usd	double	1000
hdi	double	1000
life_expectancy	double	1000
corruption_perception_index	double	1000
environmental_quality_index	double	1000
innovation_index	double	1000
quality_of_life_index	double	1000
social_freedom_index	double	1000
economic_freedom_index	double	1000
political_freedom_index	double	1000
rule_of_law_index	double	1000
corruption_index	double	1000
environmental_index	double	1000
innovation_index	double	1000
quality_of_life_index	double	1000
social_freedom_index	double	1000
economic_freedom_index	double	1000
political_freedom_index	double	1000
rule_of_law_index	double	1000

1. Filter countries where income level is 'High income'.

Code:

```
18
19 # {r}
20 # Lab 18_Task 1-1
21 # 1. Filter countries where income level is 'High income'
22 high_income_countries <- df %>%
23   filter(income_level == "High income")
24
25 # Display result for #1
26 print("High Income Countries:")
27 print(high_income_countries)
28
```

Output:

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R code from the previous block.
- Environment:** Shows the objects created: 'df' (102 obs., 24 variables) and 'high_income_countries' (38 obs., 24 variables).
- Console:** Displays the execution output, including the printed table of high-income countries.
- Table:** A preview of the 'high_income_countries' data frame.

Table Data:

country	country_code	region	income_level	college_share
Australia	AUS	ASIA-PACIFIC	High income	87.5
Austria	AUT	Europe	High income	86.7
Belgium	BEL	Europe	High income	79.8
Canada	CAN	Americas	High income	77.1
Switzerland	CHE	Europe	High income	81.1
China	CHN	Americas	High income	80.8
Croatia	HRV	Europe	High income	81.3
Czech Republic	CZE	Europe	High income	84.8
Germany	DEU	Europe	High income	77.3
Denmark	DNK	Europe	High income	82.7

Console Output:

```

> # {r}
> # Lab 18_Task 1-1
> # 1. Filter countries where income level is 'High income'
> high_income_countries <- df %>%
+   filter(income_level == "High income")
>
> # Display result for #1
> print("High Income Countries:")
[1] "High Income Countries:"
> print(high_income_countries)
# A tibble: 10 x 5
  country country_code region income_level college_share
  <chr>      <chr>      <chr>      <chr>         <dbl>
1 Australia AUS        ASIA-PACIFIC High income    87.5
2 Austria  AUT        Europe     High income    86.7
3 Belgium  BEL        Europe     High income    79.8
4 Canada   CAN        Americas   High income    77.1
5 Switzerland CHE       Europe     High income    81.1
6 China    CHN        Americas   High income    80.8
7 Croatia  HRV        Europe     High income    81.3
8 Czech Republic CZE     Europe     High income    84.8
9 Germany  DEU        Europe     High income    77.3
10 Denmark DNK        Europe     High income    82.7
  
```


2. Filter countries where region is 'Europe' or 'Americas' and life expectancy is greater than 80 years.

Code:

```
29
30+ ""{r}
31 # Lab 18_Task 1-2
32 # 2. Filter countries where region is 'Europe' or 'Americas' and life expectancy > 80
33 europe_americas_high_life_exp <- df %>%
34   filter(region %in% c("Europe", "Americas") & life_expect > 80)
35
36 # Display result for #2
37 print("Countries in Europe or Americas with Life Expectancy > 80:")
38 print(europe_americas_high_life_exp)
39
```

Output:

The screenshot shows the RStudio interface. The top-left pane contains the R code, and the bottom-left pane shows the output of the code. The output is a data frame with 10 rows and 6 columns: country, country_code, region, income_level, college_share, and inflation.

country	country_code	region	income_level	college_share	inflation
Austria	AUT	Europe	High income	48.7	3.7
Belgium	BEL	Europe	High income	48.9	1.1
Canada	CAN	Americas	High income	48.1	3.0
Switzerland	CHE	Europe	High income	46.4	0.6
Costa Rica	CRI	Americas	Upper middle income	17.7	3.8
Cyprus	CYP	Europe	High income	46.7	0.0
Germany	DEU	Europe	High income	48.1	0.1
Denmark	DNK	Europe	High income	45.1	0.7
Ireland	IRL	Europe	High income	45.1	0.4
Ireland	IRL	Europe	High income	48.1	3.0

The right-hand pane shows the Environment window with the following objects:

- df: 142 obs. of 14 variables
- europe_americas_high_life_exp: 62 obs. of 14 variables
- high_income_countries: 30 obs. of 14 variables
- income_level: 142 obs. of 1 variable

The bottom-right pane shows the console output:

```

> print("Countries in Europe or Americas with Life Expectancy > 80:")
[1] "Countries in Europe or Americas with Life Expectancy > 80:"
> print(europe_americas_high_life_exp)
# A tibble: 10 x 6
  country country_code region income_level college_share inflation
  <chr>      <chr>      <chr>      <chr>         <dbl>      <dbl>
1 Austria  AUT        Europe    High income  48.7        3.7
2 Belgium  BEL        Europe    High income  48.9        1.1
3 Canada   CAN        Americas  High income  48.1        3.0
4 Switzerland CHE      Europe    High income  46.4        0.6
5 Costa Rica CRI        Americas  Upper middle income 17.7        3.8
6 Cyprus   CYP        Europe    High income  46.7        0.0
7 Germany  DEU        Europe    High income  48.1        0.1
8 Denmark  DNK        Europe    High income  45.1        0.7
9 Ireland  IRL        Europe    High income  45.1        0.4
10 Ireland IRL        Europe    High income  48.1        3.0
  
```

3. Filter the top 5 countries based on GDP growth.

Code:

```

41
42 = ```{r}
43 # Lab 18_Task 1-3
44 # 3. Filter the top 5 countries based on GDP growth
45 top_5_gdp_growth <- df %>%
46   arrange(desc(gdp_growth)) %>%
47   head(5)
48
49 # Display result for #3
50 print("Top 5 Countries Based on GDP Growth:")
51 print(top_5_gdp_growth)
52 = ```

```

Output:

The screenshot shows an RStudio session with the following components:

- Source Editor:** Contains the R code for Lab 18_Task 1-3, specifically step 3: filtering the top 5 countries by GDP growth.
- Console:** Displays the output of the code, showing the top 5 countries based on GDP growth.
- Environment Pane:** Shows the objects created in the environment, including 'top_5_gdp_growth'.
- Files Pane:** Shows the project files, including 'Lab 18_Task 1-3.R'.

The console output shows the following data frame:

country	country_code	region	income_level	gdppercapita
Turkmenistan	TJK	Asia Pacific	Lower middle income	17.5
Burkina Faso	BF	Sub-Saharan Africa	Lower middle income	6.2
Madagascar	MG	Sub-Saharan Africa	Lower middle income	6.1
Guinea	GN	Sub-Saharan Africa	Lower middle income	4.0
Yemen	YE	Asia Pacific	Lower middle income	3.8

The Environment pane shows the following objects:

- top_5_gdp_growth:** A data frame with 5 rows and 5 columns.

4. Create a new variable named NED with a value of 1 when country is equal to Netherlands and 0 when it is not. Select the country name and NED columns side by side and arrange in descending order of NED.

Code:

```
54
55 # ""{r}
56 # Lab 1B_Task 1-4
57 # 4. Create a new variable NED (1 if country is Netherlands, otherwise 0)
58 ned_sorted <- df %>%
59   mutate(NED = ifelse(country == "Netherlands", 1, 0)) %>%
60   select(country, NED) %>%
61   arrange(desc(NED))
62
63 # Display result for #4
64 print('Netherlands Flag Variable (NED), Sorted:')
65 print(ned_sorted)
66 # ""
```

Output:

Source

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

Environment

Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment
RStudio
Global Environment

Task 2:

Use the 'countries_data.xlsx' dataset. Write code that does the following:

1. Filter for income level equal to Lower middle income equal to Education

Code:

```
67
68
69 = ```{r}
70 # Lab 18_Task 2-1
71 # 1. Filter for income level equal to 'Lower middle income'
72 lower_middle_income <- df %>%
73   filter(income_level == "Lower middle income")
74
75 # Display result for #1
76 print("Filtered for Lower Middle Income:")
77 print(lower_middle_income)
78 = ```
```

Output:

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R code from the task, with line numbers 67 through 78. The code filters the 'df' dataset for 'Lower middle income' and prints the result.
- Environment Pane:** Shows the objects in the global environment. The 'lower_middle_income' object is highlighted, showing it is a data frame with 40 rows and 24 variables.
- Console:** Displays the output of the R code. It shows the filtered data frame with columns: country_name, country_code, region, and income_level. The data includes countries like Angola, Benin, Bangladesh, and others, all categorized as 'Lower middle income'.
- Viewer Pane:** Shows a preview of the filtered data, displaying columns for country_name, country_code, region, and income_level.

2. Group by region

Code:

```

80
81 - ```{r}
82 # Lab 18_Task 2-2
83 # 2. Group by region
84 grouped_by_region <- lower_middle_income %>%
85   group_by(region)
86
87 # Display result for #2
88 print("Grouped by Region:")
89 print(grouped_by_region)
90 - ```

```

Output:

The screenshot shows the RStudio interface. The console on the left displays the following output:

```

[1] "Grouped by Region:"
# A tibble: 10 x 4
  country code region income_level
  <fct>   <fct> <fct>   <fct>
1 Angola  AGO  Middle East & Africa lower middle income
2 Benin   BEN  Middle East & Africa lower middle income
3 Bangladesh BGD  Asia South & West lower middle income
4 Botswana BOT  Middle East & Africa lower middle income
5 Cameroon CMR  Middle East & Africa lower middle income
6 Congo Kin CGO  Middle East & Africa lower middle income
7 Cuba Kin CUB  Middle East & Africa lower middle income
8 Egypt Kin EGY  Middle East & Africa lower middle income
9 Egypt Arab Rep. EGY  Middle East & Africa lower middle income
10

```

The Environment pane on the right shows the objects in the workspace:

- grouped_by_region: A tibble with 10 rows and 4 columns.
- lower_middle_income: A tibble with 10 rows and 4 columns.
- income_level: A tibble with 10 rows and 1 column.
- region: A tibble with 10 rows and 1 column.
- country: A tibble with 10 rows and 1 column.
- code: A tibble with 10 rows and 1 column.

The Data Viewer pane on the right shows the first 10 rows of the 'grouped_by_region' data frame:

country	code	region	income_level
Angola	AGO	Middle East & Africa	lower middle income
Benin	BEN	Middle East & Africa	lower middle income
Bangladesh	BGD	Asia South & West	lower middle income
Botswana	BOT	Middle East & Africa	lower middle income
Cameroon	CMR	Middle East & Africa	lower middle income
Congo Kin	CGO	Middle East & Africa	lower middle income
Cuba Kin	CUB	Middle East & Africa	lower middle income
Egypt Kin	EGY	Middle East & Africa	lower middle income
Egypt Arab Rep.	EGY	Middle East & Africa	lower middle income

3. Find (1) the median value for unemployment and (2) the number of countries in each region.

Code:

```

93 - ```{r}
94 # Lab 18 Task 2-3
95 # 3. Find median unemployment and count of countries per region
96 summary_stats <- grouped_by_region %>%
97   summarise(
98     median_unemployment = median(unemployment, na.rm = TRUE), # Median unemployment
99     num_countries = n() # Count of countries in each region
100   )
101
102 # Display result for #3
103 print("Median Unemployment & Number of Countries by Region:")
104 print(summary_stats)
105
106 - ```

```

Output:

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R code for Lab 18 Task 2-3, which calculates the median unemployment and the number of countries for each region.
- Console:** Displays the output of the R code, showing the summary_stats object as a tibble with columns region, median_unemployment, and num_countries.
- Environment:** Shows the objects created in the environment, including summary_stats, which is a tibble with 5 rows and 3 columns.
- Files:** Shows the project files, including the R script and the output file.

The output of the R code is as follows:

```

# Median Unemployment & Number of Countries by Region:
# A tibble: 5 x 3
#   region                median_unemployment num_countries
#   <fct>                  <dbl>             <dbl>
1 America                5.25                 3
2 Asia Pacific            4.25                14
3 Europe                  7.15                 3
4 Middle East & Africa    5.95                11
5 Oceania                 5.25                 3

```

4. Sort the results from the highest median optimal level downward.

Code:

```

108
109 - ```{r}
110 # Lab 18_Task 2-4
111 # 4. Sort from highest to lowest median unemployment
112 sorted_result <- summary_stats %>%
113   arrange(desc(median_unemployment))
114
115 # Display result for #4
116 print("Sorted by Highest Median Unemployment:")
117 print(sorted_result)
118 - ```

```

Output:

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R code for Lab 18_Task 2-4, specifically the sorting operation.
- Environment:** Shows the objects created in the environment, including 'summary_stats' (a tibble with 140 rows and 24 variables) and 'sorted_result' (a tibble with 4 rows and 3 variables).
- Console:** Displays the output of the R code, showing the sorted result as a table.
- Table View:** A visual representation of the 'sorted_result' tibble, showing the first 4 rows.

Sorted Result Table:

region	median_unemployment	work_provision
Africa	4.25	12
America	3.25	3
Europe	2.15	2
Middle East & North Africa	0.25	1

Task 3:

Suppose we work for a housing developer like Toll Brothers (NYSE: TOL) and want to allocate resources to marketing and financing the building of luxury homes in major US metropolitan areas. We have data for a test market in the file 'hprice.csv'.

Do the following:

Code:

```
119
120
121 - ```{r}
122   # Load required libraries
123   library(tidyverse)
124   library(kableExtra)
125 - ```
126
127
128 - ```{r}
129   # Lab 18_Task 3
130   # Load the dataset
131   hprice <- read.csv("hprice.csv")
132 - ```
133
```


2. Group the Pivot table data by Brick and Neighborhood.

Code:

```
145  
146 + ```{r}  
147 # Lab 18_Task 3-2  
148 # Group the data by Brick and Neighborhood  
149 grouped_data <- high_value_properties %>%  
150   group_by(Brick, Neighborhood)  
151 + ```
```

3. Fetch the Average Price, Average Square Foot, and Price Per Square Foot

Code:

```

i4+ ""{r}
i5 # Lab 18_Task 3-3
i6 # Calculate Average Price, Average Square Foot, and Price Per Square Foot
i7 summary_data <- grouped_data %>%
i8   summarise(
i9     Avg_Price = mean(Price),
i10    Avg_SqFt = mean(SqFt),
i11    Price_Per_SqFt = mean(Price / SqFt)
i12  )
i13
i4+ ""

```

Output:

The screenshot shows the RStudio interface. The script editor on the left contains the following code:

```

# Lab 18_Task 3-3
# Calculate Average Price, Average Square Foot, and Price Per Square Foot
summary_data <- grouped_data %>%
  summarise(
    Avg_Price = mean(Price),
    Avg_SqFt = mean(SqFt),
    Price_Per_SqFt = mean(Price / SqFt)
  )

```

The console on the right shows the output of the code, which is a data frame with 10 rows and 4 columns: Price, SqFt, Price_Per_SqFt, and Neighborhood. The data is as follows:

Price	SqFt	Price_Per_SqFt	Neighborhood
8.400	100	0.084	Mar 11, 2015, 5:08 PM
8.400	100	0.084	Mar 11, 2015, 5:08 PM
32.200	100	0.322	Mar 11, 2015, 5:08 PM
41.000	100	0.410	Mar 11, 2015, 5:08 PM
40.000	100	0.400	Mar 11, 2015, 5:08 PM
3.500	100	0.035	Mar 11, 2015, 5:08 PM
3.000	100	0.030	Mar 11, 2015, 5:08 PM
3.000	100	0.030	Mar 11, 2015, 5:08 PM
3.000	100	0.030	Mar 11, 2015, 5:08 PM
3.000	100	0.030	Mar 11, 2015, 5:08 PM

4. Visualize 1-3 in a table (Use the kableExtra package and knitr:: command)

Code:

```

165
166
167 # {r}
168 # Lab 18_Task 3-4
169 # Visualize the data in a table
170 kable(summary_data, format = "html", caption = "Summary of High-Value Neighborhoods") %>%
171   kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover", "condensed"))
172

```

Output:

The screenshot shows an RStudio window with a script editor on the left and a viewer pane on the right. The script editor contains R code that uses the `kableExtra` and `knitr` packages to create a styled HTML table. The code is as follows:

```

# {r}
# Lab 18_Task 3-4
# Visualize the data in a table
kable(summary_data, format = "html", caption = "Summary of High-Value Neighborhoods") %>%
  kable_styling(full_width = FALSE, bootstrap_options = c("striped", "hover", "condensed"))

```

The viewer pane displays the resulting HTML table, which is titled "Summary of High-Value Neighborhoods". The table has the following structure:

Area	Neighborhood	Avg Price	Avg SqFt	Price Per SqFt
No	East	1177000	1001.00	1175.30
No	North	2000000	1020.10	1960.70
No	West	1600000	1010.00	1584.16
Yes	East	1000000	1001.00	1000.00
Yes	North	1100000	1010.00	1089.11
Yes	West	1100000	1020.10	1078.20

Task 4:**Gen AI Prompt Engineering:**

You work as a data analyst for a retail company. The marketing department has provided you with customer transaction data in an Excel file, `customer_sales.csv`, and wants you to perform customer segmentation analysis based on how much customers spend and calculate potential impact of targeted marketing campaigns. Use spending-based customer classification system that segments customers into three categories:

"High Spender" - Customers who spent \$300 or more

"Medium Spender" - Customers who spent between \$150 and \$299.99

"Low Spender" - Customers who spent less than \$150

Bot Used: Claude.AI**Prompt:**

The screenshot shows a chat window with a dark background. On the left, there is a 'Task Requirements' box with a list of 6 steps: 1. Read the provided CSV file (customer_sales.csv) into R, 2. Clean and analyze the data, 3. Segment customers based on spending levels, 4. Create multiple analysis perspectives by product, age group, and location, 5. Track monthly sales trends, 6. Export all analysis into a single R script for multiple reuse. Below this is an 'Expected Output' section listing 7 items: 1. Customer Spending: Shows each customer's total spending, average purchase amount, number of purchases, and spending category, 2. Product Categories: Analyzes sales performance by product category, 3. Age Group Analysis: Breaks down spending by age groups and gender, 4. Location Analysis: Shows spending patterns across different locations, 5. Monthly Sales: Displays monthly revenue and its seasonality, 6. Marketing Impact, 7. Customer Recommendations. To the right of the requirements is a file icon labeled 'customer_sales.csv'. The main chat area contains a prompt starting with 'You work as a data analyst for a retail company...' and ends with 'Give me detailed steps to perform this in R'. An 'Edit' button is at the bottom right.

Task Requirements

1. Read the provided CSV file (customer_sales.csv) into R
2. Clean and analyze the data
3. Segment customers based on spending levels
4. Create multiple analysis perspectives by product, age group, and location
5. Track monthly sales trends
6. Export all analysis into a single R script for multiple reuse

Expected Output

After running R code, R code generates a new R script that contains analysis results with the following sections:

1. Customer Spending: Shows each customer's total spending, average purchase amount, number of purchases, and spending category
2. Product Categories: Analyzes sales performance by product category
3. Age Group Analysis: Breaks down spending by age groups and gender
4. Location Analysis: Shows spending patterns across different locations
5. Monthly Sales: Displays monthly revenue and its seasonality
6. Marketing Impact
7. Customer Recommendations

customer_sales.csv

Prompt:

You work as a data analyst for a retail company. The marketing department has provided you with customer transaction data in an Excel file, `customer_sales.csv`, and wants you to perform customer segmentation analysis based on how much customers spend and calculate potential impact of targeted marketing campaigns. Use spending-based customer classification system that segments customers into three categories: "High Spender" - Customers who spent \$300 or more "Medium Spender" - Customers who spent between \$150 and \$299.99 "Low Spender" - Customers who spent less than \$150

Give me detailed steps to perform this in R

Edit

Bot:

Clouds Customer Segmentation Analysis by Spending Share Star Close

It helps you create an AI script to perform customer segmentation analysis based on your requirements. Here's a detailed step-by-step approach:

`python` **Customer Segmentation Analysis by Spending**
[View script](#)

Explanation of the Customer Segmentation Analysis Script

This script follows all the requirements specified in your task and implements the required customer segmentation based on spending levels. Here's a breakdown of what each function does:

1. Reading and Preparing the Data

- Ensures necessary libraries (pandas, numpy) are installed and imported for data manipulation and analysis.
- Reads the customer purchase history file.
- Converts purchase dates to proper Date format.
- Checks for and reports any missing values in the dataset.

2. Customer Segmentation

- Calculates total spending, average purchase count, and average purchase value for each customer.
- Segments customers into three categories as requested:
 - High Spender (\$500 or more)
 - Medium Spender (\$20 to \$499.99)

[Return to Clouds](#)

Clouds ID: 10001 15 [View script](#)

```

import pandas as pd
import numpy as np
import sys
import os

# Step 1: Read the CSV file
def read_data(file_path):
    """Reads the customer purchase history CSV file and returns a DataFrame"""
    df = pd.read_csv(file_path)

    # Step 2: Check for missing values
    missing_data = df.isnull().sum()
    print("Missing values in each column:")
    print(missing_data)

    # Step 3: Convert purchase dates to proper Date format
    df['Purchase Date'] = pd.to_datetime(df['Purchase Date'])

    # Step 4: Calculate total spending, average purchase count, and average purchase value
    df['Total Spending'] = df['Purchase Price'] * df['Quantity']
    df['Avg Purchase Count'] = df['Purchase Price'] / df['Purchase Price']
    df['Avg Purchase Value'] = df['Purchase Price'] * df['Quantity']

    # Step 5: Segment customers into three categories as requested
    df['Customer Segment'] = df['Total Spending'].apply(lambda x: 'High Spender' if x >= 500 else 'Medium Spender' if x > 20 else 'Low Spender')

    # Step 6: Save the segmented data to a new CSV file
    df.to_csv('customer_segmentation.csv', index=False)

    # Step 7: Print the results
    print("Customer Segmentation Results:")
    print(df.groupby('Customer Segment').agg({'Total Spending': 'sum', 'Avg Purchase Count': 'mean', 'Avg Purchase Value': 'mean'}))

# Step 8: Main function
def main():
    file_path = 'customer_purchase_history.csv'
    read_data(file_path)

# Step 9: Run the main function
if __name__ == '__main__':
    main()
  
```

[View script](#) 15 [View script](#)

Clouds Customer Segmentation Analysis by Spending Share Star Close

It helps you create an AI script to perform customer segmentation analysis based on your requirements. Here's a detailed step-by-step approach:

`python` **Customer Segmentation Analysis by Spending**
[View script](#)

Explanation of the Customer Segmentation Analysis Script

This script follows all the requirements specified in your task and implements the required customer segmentation based on spending levels. Here's a breakdown of what each function does:

1. Reading and Preparing the Data

- Ensures necessary libraries (pandas, numpy) are installed and imported for data manipulation and analysis.
- Reads the customer purchase history file.
- Converts purchase dates to proper Date format.
- Checks for and reports any missing values in the dataset.

2. Customer Segmentation

- Calculates total spending, average purchase count, and average purchase value for each customer.
- Segments customers into three categories as requested:
 - High Spender (\$500 or more)
 - Medium Spender (\$20 to \$499.99)
 - Low Spender (Less than \$20)

3. Additional Analysis

- Adds demographic information (age, gender, location) to the segmented spending data.
- Creates age groups for more granular analysis.

4. Multiple Analysis Perspectives

- Product Category Analysis:** Calculates average transactions and average purchase count by product category.
- Age Group Analysis:** Breaks down spending patterns by age groups and gender.
- Location Analysis:** Analyzes spending patterns across a chain, suburban, and rural locations.
- Monthly Sales Trends:** Tracks revenue and transaction counts by month.

5. Marketing Impact and Recommendations

- Generates potential revenue impact of targeted marketing campaigns based on:
 - Different conversion rates for each age group category.
 - Estimated additional purchase amount by segment.

[Return to Clouds](#)

Clouds ID: 10001 15 [View script](#)

```

import pandas as pd
import numpy as np
import sys
import os

# Step 1: Read the CSV file
def read_data(file_path):
    """Reads the customer purchase history CSV file and returns a DataFrame"""
    df = pd.read_csv(file_path)

    # Step 2: Check for missing values
    missing_data = df.isnull().sum()
    print("Missing values in each column:")
    print(missing_data)

    # Step 3: Convert purchase dates to proper Date format
    df['Purchase Date'] = pd.to_datetime(df['Purchase Date'])

    # Step 4: Calculate total spending, average purchase count, and average purchase value
    df['Total Spending'] = df['Purchase Price'] * df['Quantity']
    df['Avg Purchase Count'] = df['Purchase Price'] / df['Purchase Price']
    df['Avg Purchase Value'] = df['Purchase Price'] * df['Quantity']

    # Step 5: Segment customers into three categories as requested
    df['Customer Segment'] = df['Total Spending'].apply(lambda x: 'High Spender' if x >= 500 else 'Medium Spender' if x > 20 else 'Low Spender')

    # Step 6: Save the segmented data to a new CSV file
    df.to_csv('customer_segmentation.csv', index=False)

    # Step 7: Print the results
    print("Customer Segmentation Results:")
    print(df.groupby('Customer Segment').agg({'Total Spending': 'sum', 'Avg Purchase Count': 'mean', 'Avg Purchase Value': 'mean'}))

# Step 8: Main function
def main():
    file_path = 'customer_purchase_history.csv'
    read_data(file_path)

# Step 9: Run the main function
if __name__ == '__main__':
    main()
  
```

[View script](#) 15 [View script](#)

[illegible]

The screenshot displays a Jupyter Notebook with the following content:

```

1 # Import necessary libraries
2 import pandas as pd
3 import numpy as np
4
5 # Load data
6 # Load customer spending data
7 customer_spending = pd.read_csv('customer_spending.csv')
8
9 # Load customer demographic data
10 customer_demographic = pd.read_csv('customer_demographic.csv')
11
12 # Merge data
13 customer_data = pd.merge(customer_spending, customer_demographic, on='customer_id')
14
15 # Filter data
16 # Filter for customers who have spent more than $100
17 customer_data = customer_data[customer_data['amount'] > 100]
18
19 # Group data
20 # Group by customer_id and calculate total spending
21 customer_data.groupby('customer_id').sum().reset_index(inplace=True)
22
23 # Sort data
24 # Sort by total spending in descending order
25 customer_data.sort_values(by='amount', ascending=False, inplace=True)
26
27 # Display data
28 # Display the first 10 rows
29 customer_data.head(10)

```

The output of the notebook shows a table with 10 rows of customer spending data:

customer_id	date	amount	customer_name
1	2023-01-01	150.00	John Doe
2	2023-01-02	120.00	Jane Smith
3	2023-01-03	90.00	Bob Johnson
4	2023-01-04	80.00	Alice Brown
5	2023-01-05	70.00	Charlie Davis
6	2023-01-06	60.00	Diana Prince
7	2023-01-07	50.00	Edward Nigma
8	2023-01-08	40.00	Fiona Gale
9	2023-01-09	30.00	George Clooney
10	2023-01-10	20.00	Helen Mirren

The screenshot displays a Jupyter Notebook with a code cell on the left and its output on the right. The code cell contains a script that reads a CSV file, performs a location analysis, and saves the results. The output cell shows a table of results with columns for location, date, and time.

```

1 # Import multiple analysis components
2 # Import analysis module
3 # Import pandas module
4 # Import datetime module
5 # Import location module
6 # Import results module
7 # Import save module
8 # Import load module
9 # Import plot module
10 # Import save module
11 # Import load module
12 # Import plot module
13 # Import save module
14 # Import load module
15 # Import plot module
16 # Import save module
17 # Import load module
18 # Import plot module
19 # Import save module
20 # Import load module
21 # Import plot module
22 # Import save module
23 # Import load module
24 # Import plot module
25 # Import save module
26 # Import load module
27 # Import plot module
28 # Import save module
29 # Import load module
30 # Import plot module
31 # Import save module
32 # Import load module
33 # Import plot module
34 # Import save module
35 # Import load module
36 # Import plot module
37 # Import save module
38 # Import load module
39 # Import plot module
40 # Import save module
41 # Import load module
42 # Import plot module
43 # Import save module
44 # Import load module
45 # Import plot module
46 # Import save module
47 # Import load module
48 # Import plot module
49 # Import save module
50 # Import load module
51 # Import plot module
52 # Import save module
53 # Import load module
54 # Import plot module
55 # Import save module
56 # Import load module
57 # Import plot module
58 # Import save module
59 # Import load module
60 # Import plot module
61 # Import save module
62 # Import load module
63 # Import plot module
64 # Import save module
65 # Import load module
66 # Import plot module
67 # Import save module
68 # Import load module
69 # Import plot module
70 # Import save module
71 # Import load module
72 # Import plot module
73 # Import save module
74 # Import load module
75 # Import plot module
76 # Import save module
77 # Import load module
78 # Import plot module
79 # Import save module
80 # Import load module
81 # Import plot module
82 # Import save module
83 # Import load module
84 # Import plot module
85 # Import save module
86 # Import load module
87 # Import plot module
88 # Import save module
89 # Import load module
90 # Import plot module
91 # Import save module
92 # Import load module
93 # Import plot module
94 # Import save module
95 # Import load module
96 # Import plot module
97 # Import save module
98 # Import load module
99 # Import plot module
100 # Import save module

```

The output cell shows a table of results with columns for location, date, and time. The table contains 10 rows of data, including locations like 'New York City', 'Los Angeles', and 'San Francisco'.

Location	Date	Time
New York City	2015-01-01	10:00 AM
Los Angeles	2015-01-01	11:00 AM
San Francisco	2015-01-01	12:00 PM
New York City	2015-01-01	13:00 PM
Los Angeles	2015-01-01	14:00 PM
San Francisco	2015-01-01	15:00 PM
New York City	2015-01-01	16:00 PM
Los Angeles	2015-01-01	17:00 PM
San Francisco	2015-01-01	18:00 PM
New York City	2015-01-01	19:00 PM

[illegible][illegible]

The screenshot displays a Jupyter Notebook interface with a code editor on the left and an output window on the right. The code in the notebook includes:

```

# Import all analysis from a single file with wildcard syntax
# Create notebook
nb = createNotebook()

# Add worksheets
addWorksheet(nb, 'Customer Spending')
addWorksheet(nb, 'Product Categorization')
addWorksheet(nb, 'Age Group Analysis')
addWorksheet(nb, 'Weekly Sales')
addWorksheet(nb, 'Marketing Impact')

# Save notebook
saveNotebook(nb, 'Customer_Analysis_Results.xlsx', overwrite = True)

# Print local file location
file_path = file.getPath()
print('Analysis completed. Results exported to:', file_path)

```

The output window on the right shows a list of files and their sizes, including:

File Name	Size	Modified
Customer_Analysis_Results.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM
Product_Categorization.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM
Age_Group_Analysis.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM
Weekly_Sales.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM
Marketing_Impact.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM

The screenshot displays a Jupyter Notebook interface with a code editor on the left and an output window on the right. The code in the notebook includes:

```

# Import all analysis from a single file with wildcard syntax
# Create notebook
nb = createNotebook()

# Add worksheets
addWorksheet(nb, 'Customer Spending')
addWorksheet(nb, 'Product Categorization')
addWorksheet(nb, 'Age Group Analysis')
addWorksheet(nb, 'Weekly Sales')
addWorksheet(nb, 'Marketing Impact')

# Save notebook
saveNotebook(nb, 'Customer_Analysis_Results.xlsx', overwrite = True)

# Print local file location
file_path = file.getPath()
print('Analysis completed. Results exported to:', file_path)

```

The output window on the right shows a list of files and their sizes, including:

File Name	Size	Modified
Customer_Analysis_Results.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM
Product_Categorization.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM
Age_Group_Analysis.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM
Weekly_Sales.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM
Marketing_Impact.xlsx	10.0 KB	Mar 17, 2023, 11:00 AM

This the excel we got after running R script.

product_id	unit_weight	average_price	max_price	min_price	shipping_weight	avg_price	location	avg_price
Product A	100	100.14	100.14	100.14	100.14	100.14	Product A	100.14
Product B	100	100.14	100.14	100.14	100.14	100.14	Product B	100.14
Product C	100	100.14	100.14	100.14	100.14	100.14	Product C	100.14
Product D	100	100.14	100.14	100.14	100.14	100.14	Product D	100.14
Product E	100	100.14	100.14	100.14	100.14	100.14	Product E	100.14
Product F	100	100.14	100.14	100.14	100.14	100.14	Product F	100.14
Product G	100	100.14	100.14	100.14	100.14	100.14	Product G	100.14
Product H	100	100.14	100.14	100.14	100.14	100.14	Product H	100.14
Product I	100	100.14	100.14	100.14	100.14	100.14	Product I	100.14
Product J	100	100.14	100.14	100.14	100.14	100.14	Product J	100.14

product_id	unit_weight	avg_price	max_price	avg_price
Product A	100	100.14	100.14	100.14
Product B	100	100.14	100.14	100.14
Product C	100	100.14	100.14	100.14
Product D	100	100.14	100.14	100.14
Product E	100	100.14	100.14	100.14

Windows • [Icons] [Address Bar: 1000001_201905_001.xlsx] [Search: 1000001_201905_001.xlsx] [Buttons: Undo, Redo, Save, Print, etc.]

File Home Insert Layout Design Formulas Data Review View Developer [Buttons: Font, Paragraph, Styles, etc.]

1000001_201905_001

Age_Group	Gender	Total_Persons	Male_Persons	Female_Persons	Age_Group
1-17	F	1000000	0	1000000	1
1-17	M	1000000	1000000	0	2
18-24	F	1000000	0	1000000	3
18-24	M	1000000	1000000	0	4
25-34	F	1000000	0	1000000	5
25-34	M	1000000	1000000	0	6
35-44	F	1000000	0	1000000	7
35-44	M	1000000	1000000	0	8
45-54	F	1000000	0	1000000	9
45-54	M	1000000	1000000	0	10

1000001_201905_001

Windows • [Icons] [Address Bar: 1000001_201905_001.xlsx] [Search: 1000001_201905_001.xlsx] [Buttons: Undo, Redo, Save, Print, etc.]

File Home Insert Layout Design Formulas Data Review View Developer [Buttons: Font, Paragraph, Styles, etc.]

1000001_201905_001

Age_Group	Gender	Total_Persons	Male_Persons	Female_Persons	Age_Group
1-17	F	1000000	0	1000000	1
1-17	M	1000000	1000000	0	2
18-24	F	1000000	0	1000000	3
18-24	M	1000000	1000000	0	4
25-34	F	1000000	0	1000000	5
25-34	M	1000000	1000000	0	6
35-44	F	1000000	0	1000000	7
35-44	M	1000000	1000000	0	8
45-54	F	1000000	0	1000000	9
45-54	M	1000000	1000000	0	10

1000001_201905_001

Windows • [Icons] [Address Bar: 1000001_201905_001.xlsx] [Search: 1000001_201905_001.xlsx] [Buttons: Undo, Redo, Save, Print, etc.]

File Home Insert Layout Design Formulas Data Review View Developer [Buttons: Font, Paragraph, Styles, etc.]

1000001_201905_001

Age_Group	Gender	Total_Persons	Male_Persons	Female_Persons	Age_Group
1-17	F	1000000	0	1000000	1
1-17	M	1000000	1000000	0	2
18-24	F	1000000	0	1000000	3
18-24	M	1000000	1000000	0	4
25-34	F	1000000	0	1000000	5
25-34	M	1000000	1000000	0	6
35-44	F	1000000	0	1000000	7
35-44	M	1000000	1000000	0	8
45-54	F	1000000	0	1000000	9
45-54	M	1000000	1000000	0	10

1000001_201905_001

Windows • [Icons] [Address Bar: 1000001_201905_001.xlsx] [Search: 1000001_201905_001.xlsx] [Buttons: Undo, Redo, Save, Print, etc.]

File Home Insert Layout Design Formulas Data Review View Developer [Buttons: Font, Paragraph, Styles, etc.]

1000001_201905_001

Age_Group	Gender	Total_Persons	Male_Persons	Female_Persons	Age_Group
1-17	F	1000000	0	1000000	1
1-17	M	1000000	1000000	0	2
18-24	F	1000000	0	1000000	3
18-24	M	1000000	1000000	0	4
25-34	F	1000000	0	1000000	5
25-34	M	1000000	1000000	0	6
35-44	F	1000000	0	1000000	7
35-44	M	1000000	1000000	0	8
45-54	F	1000000	0	1000000	9
45-54	M	1000000	1000000	0	10

1000001_201905_001

IE5390 Structured Data Analytics

The screenshot shows the Microsoft Excel application window. The title bar indicates the file name is 'Accounting_Excel_101.xlsx'. The ribbon is set to the 'Formulas' tab, with the 'Calculate Now' button highlighted. The spreadsheet contains the following data:

entry_id	entry_name	entry_type	entry_status	entry_date
1001	John Doe	Employee	Active	2023-01-01
1002	Jane Smith	Employee	Active	2023-01-01
1003	Bob Johnson	Employee	Inactive	2023-01-01
1004	Alice Brown	Employee	Active	2023-01-01
1005	Charlie Davis	Employee	Active	2023-01-01
1006	Diana Prince	Employee	Active	2023-01-01
1007	Ethan Hunt	Employee	Active	2023-01-01
1008	Fiona Glenanne	Employee	Active	2023-01-01
1009	Gavin Hastings	Employee	Active	2023-01-01
1010	Helen Mirren	Employee	Active	2023-01-01
1011	Ian McKellen	Employee	Active	2023-01-01
1012	Judi Dench	Employee	Active	2023-01-01
1013	Kate Winslet	Employee	Active	2023-01-01
1014	Leonardo DiCaprio	Employee	Active	2023-01-01
1015	Mel Gibson	Employee	Active	2023-01-01
1016	Natalie Portman	Employee	Active	2023-01-01
1017	Orlando Bloom	Employee	Active	2023-01-01
1018	Peter Dinklage	Employee	Active	2023-01-01
1019	Quentin Tarantino	Employee	Active	2023-01-01
1020	Rachel Watson	Employee	Active	2023-01-01
1021	Samuel L. Jackson	Employee	Active	2023-01-01
1022	Tina Turner	Employee	Active	2023-01-01
1023	Uma Thurman	Employee	Active	2023-01-01
1024	Viggo Mortensen	Employee	Active	2023-01-01
1025	Wendie Renner	Employee	Active	2023-01-01
1026	Xavier Dolan	Employee	Active	2023-01-01
1027	Yasmine Picaud	Employee	Active	2023-01-01
1028	Zoe Lister-Jones	Employee	Active	2023-01-01

The screenshot shows an Excel spreadsheet titled "Inventory" with the following data:

Item	Item Name	Item Price	Item Quantity	Item Total	Item Tax	Item Grand Total
Item 1	Item 1	100.00	10	1000.00	100.00	1100.00
Item 2	Item 2	200.00	5	1000.00	100.00	1100.00
Item 3	Item 3	300.00	3	900.00	90.00	990.00

item	brand	category	price	description
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000
1000	1000	1000	1000	1000 1000 1000 1000 1000

- As we can see the bot's query resulted in a clear response and we have also achieved the required result.