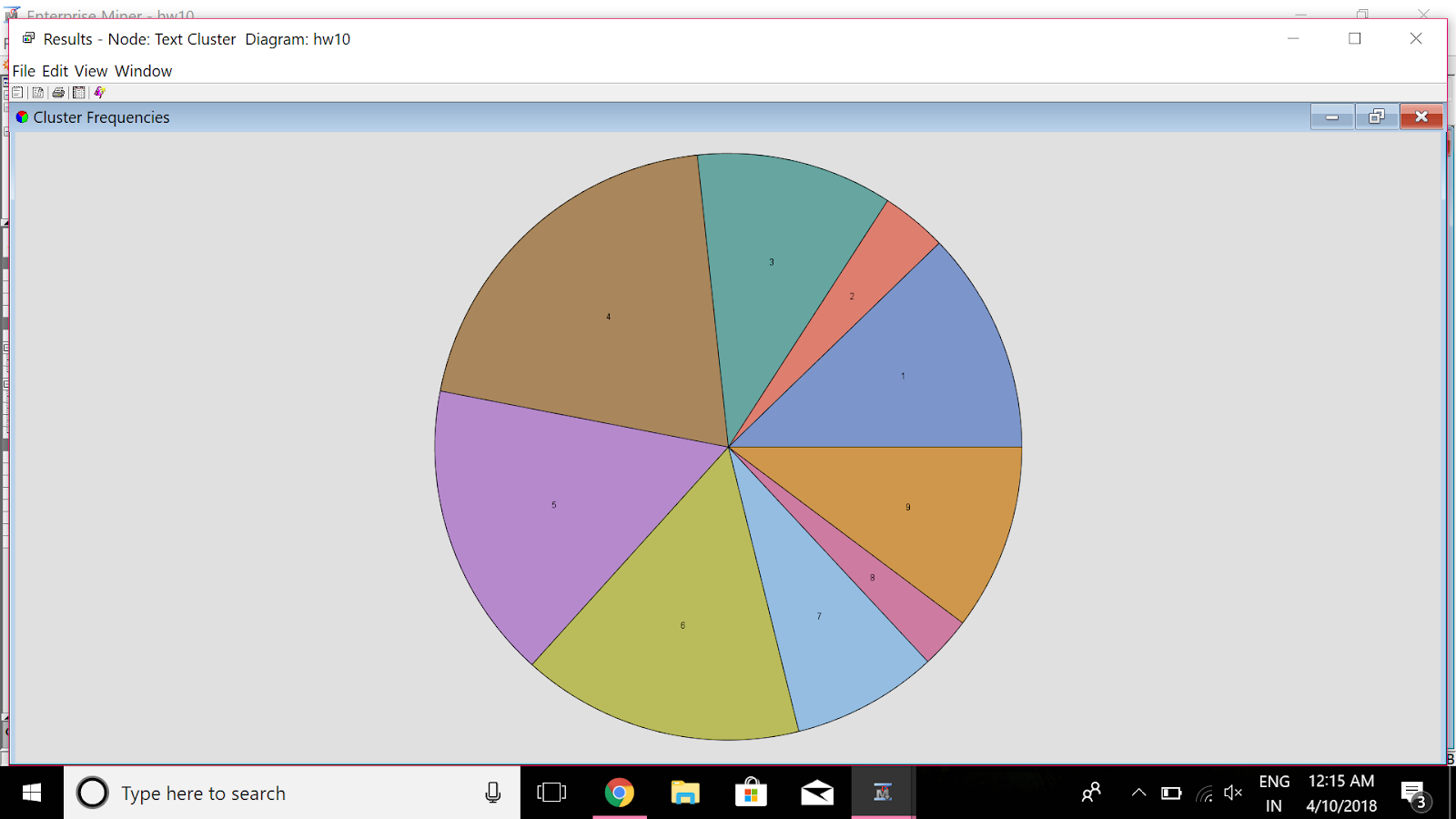
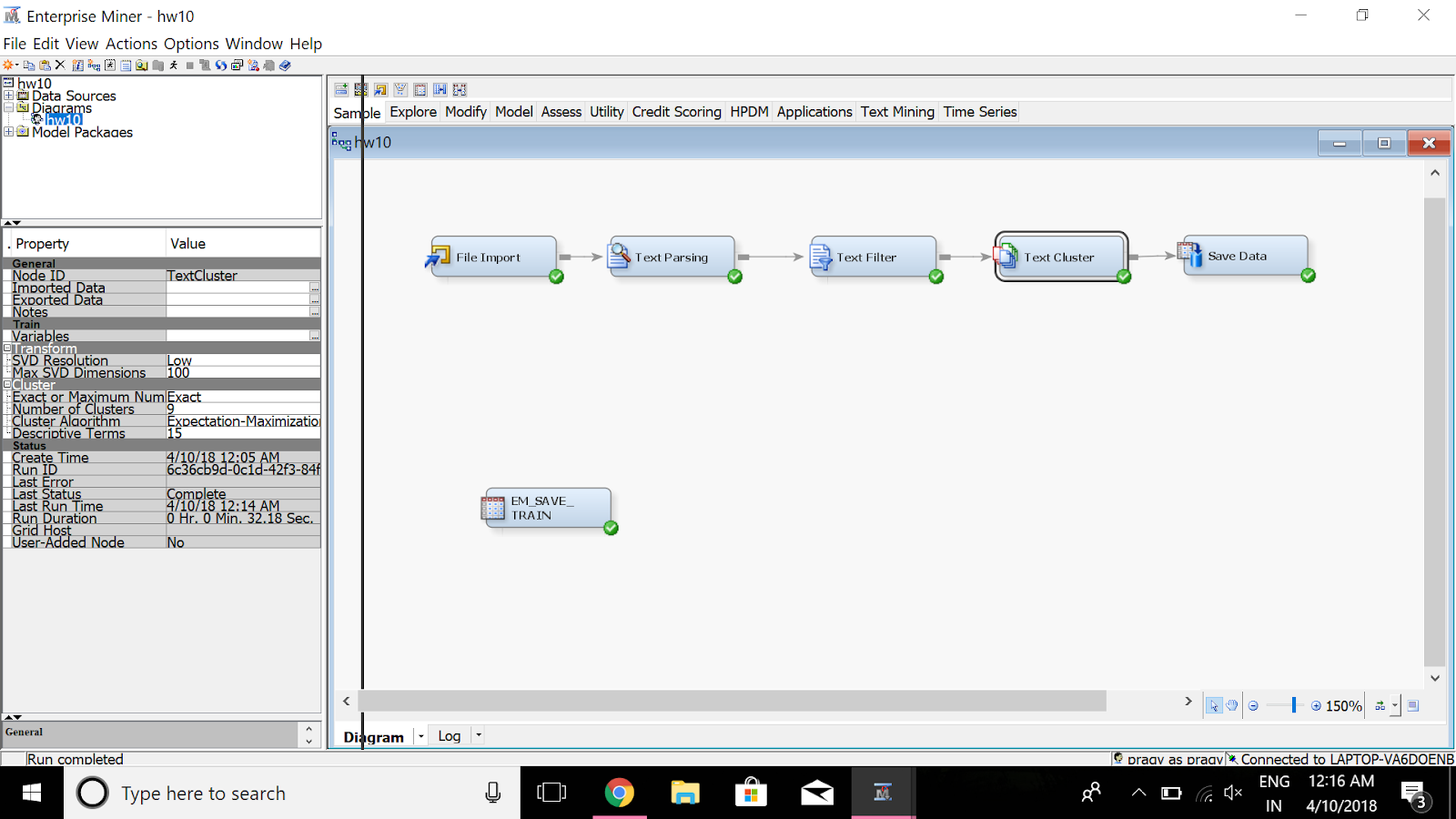
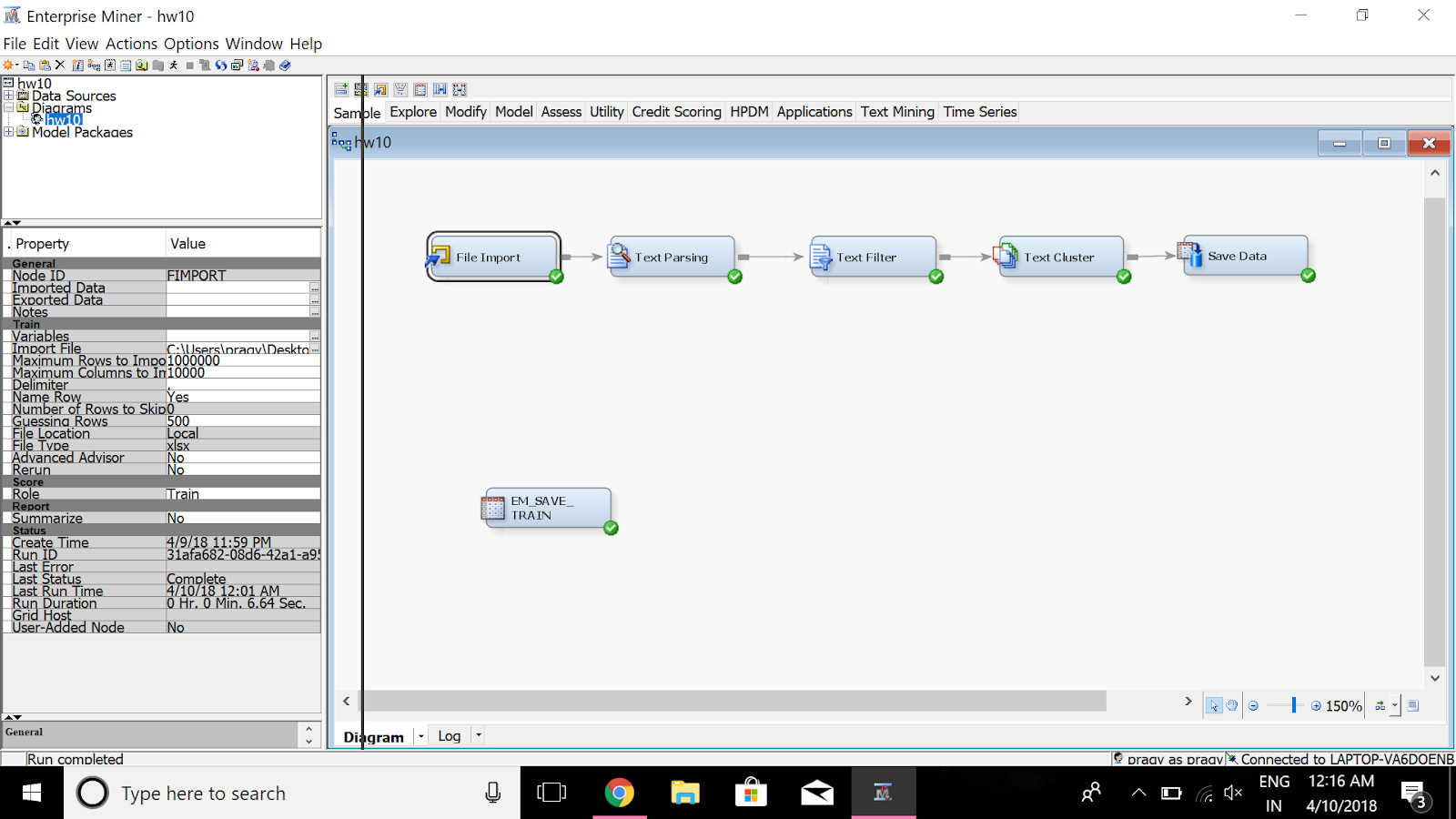
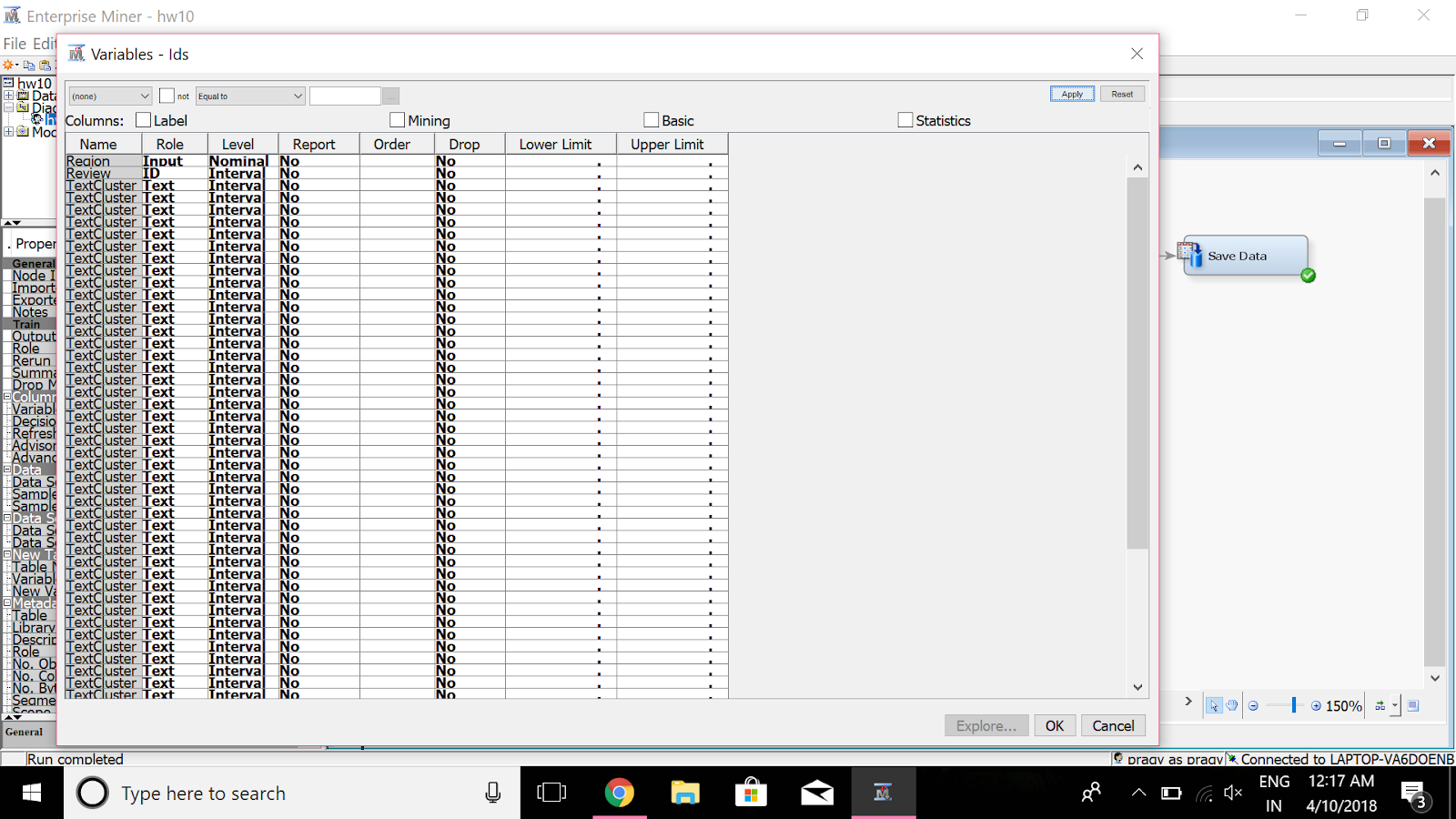
I have tried the assignment in both SAS and Python.

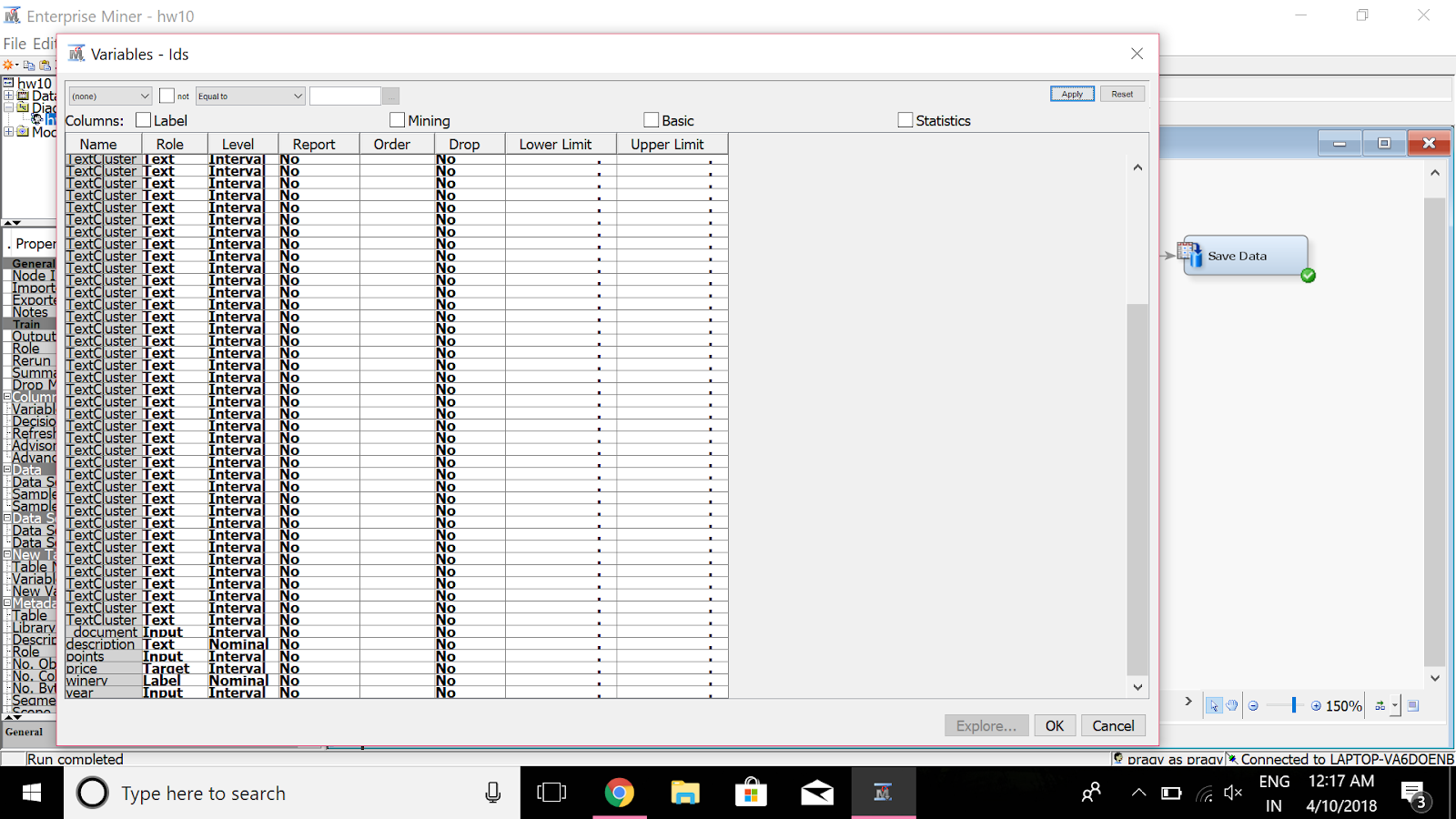
**SAS**

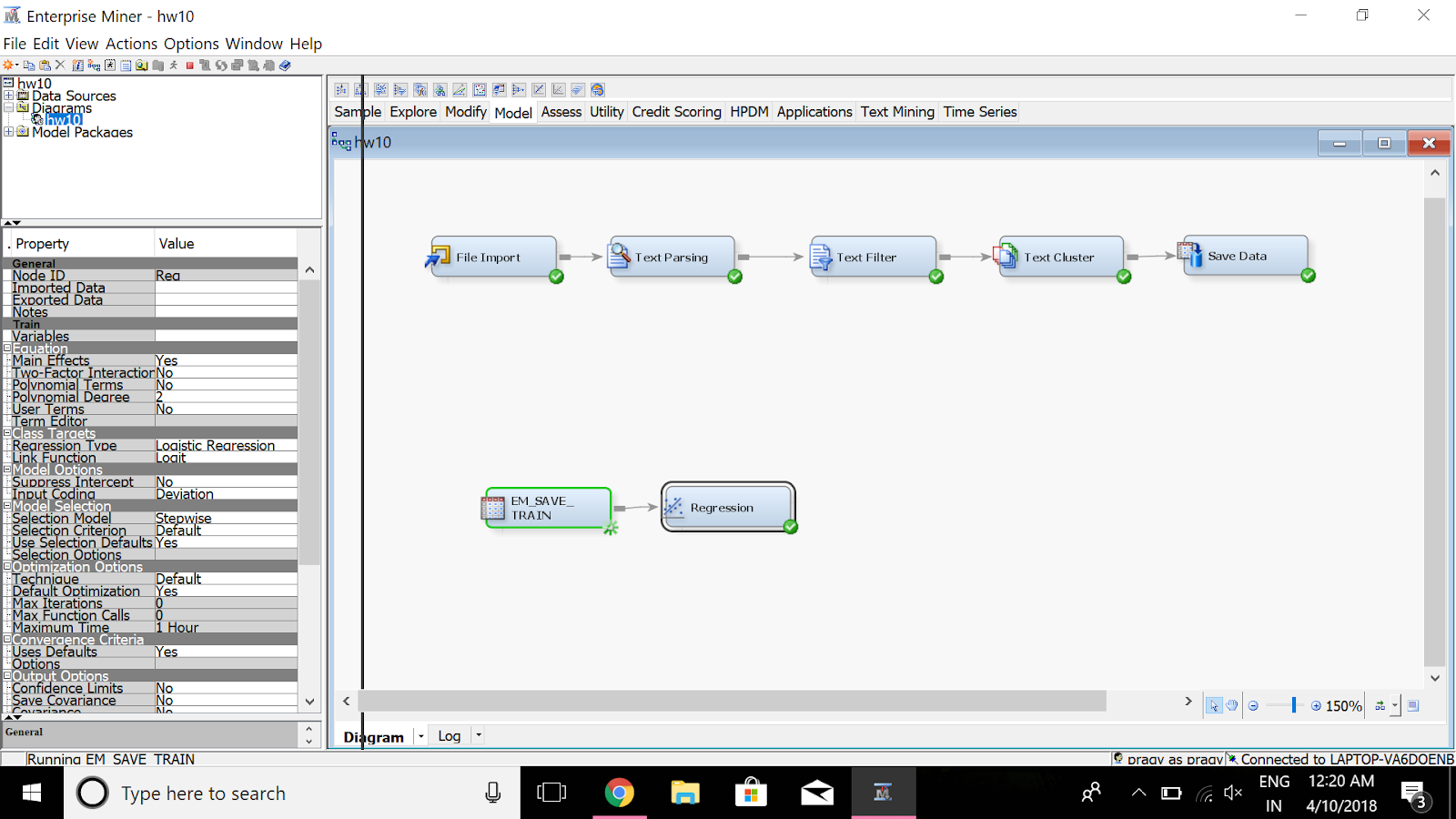












 Analysis of Variance

                                  Sum of

Source                 DF Squares Mean Square    F Value Pr > F

Model                  39 5201467     133371 117.20 <.0001

Error                5609 6382649    1137.929910

Corrected Total      5648 11584116

             Model Fit Statistics

R-Square        0.4490 Adj R-Sq        0.4452

AIC         39791.6788   BIC 39794.3113

SBC         40057.2481   C(p) 35.6781

                 Type 3 Analysis of Effects

                                   Sum of

Effect                    DF Squares F Value    Pr > F

Region                    16 592151.692 32.52    <.0001

TextCluster\_SVD11          1 22472.5282 19.75   <.0001

TextCluster\_SVD12          1 155321.325 136.49   <.0001

TextCluster\_SVD13          1 29869.2704 26.25   <.0001

TextCluster\_SVD14          1 7260.7682 6.38   0.0116

TextCluster\_SVD15          1 8669.3107 7.62   0.0058

TextCluster\_SVD16          1 12248.8335 10.76   0.0010

TextCluster\_SVD19          1 5243.6527 4.61   0.0319

TextCluster\_SVD2           1 29937.3103 26.31   <.0001

TextCluster\_SVD20          1 22753.0170 20.00   <.0001

TextCluster\_SVD21          1 55064.6377 48.39   <.0001

TextCluster\_SVD23          1 23934.4866 21.03   <.0001

TextCluster\_SVD24          1 21108.9920 18.55   <.0001

TextCluster\_SVD30          1 41099.7896 36.12   <.0001

TextCluster\_SVD32          1 20443.0418 17.97   <.0001

TextCluster\_SVD39          1 6250.1802 5.49   0.0191

TextCluster\_SVD4           1 80277.8117 70.55   <.0001

TextCluster\_SVD41          1 4878.4134 4.29   0.0384

TextCluster\_SVD45          1 5186.3524 4.56   0.0328

TextCluster\_SVD5           1 85720.1918 75.33   <.0001

TextCluster\_SVD6           1 4649.4178 4.09   0.0433

TextCluster\_SVD8           1 33321.7904 29.28   <.0001

TextCluster\_SVD9           1 18171.1986 15.97   <.0001

points                     1 1433536.61 1259.78   <.0001

                           Analysis of Maximum Likelihood Estimates

                                                                 Standard

Parameter                                       DF Estimate Error t Value Pr > |t|

Intercept                                        1 -490.8 15.1580 -32.38 <.0001

Region               California Other       1 -5.1178 3.4856   -1.47 0.1421

Region               Central Coast       1 -0.9603 3.2287   -0.30 0.7662

Region               Central Valley       1 -8.8512 4.3218   -2.05 0.0406

Region               Clear Lake       1 10.3801 31.9057    0.33 0.7449

Region               High Valley       1 -2.7591 18.5902   -0.15 0.8820

Region               Lake County       1 -3.9944 6.1981   -0.64 0.5193

Region               Mendocino       1 -7.2383 6.5186   -1.11 0.2669

Region               Mendocino County       1 -12.1969 6.7002   -1.82 0.0688

Region               Mendocino Ridge       1 9.7025 18.5918    0.52 0.6018

Region               Mendocino/Lake Counties     0 0 .     . .

Region               Napa       1 21.9288 3.0958    7.08 <.0001

Region               Napa-Sonoma       1 12.2873 5.9743    2.06 0.0398

Region               North Coast       1 -9.2420 4.5830   -2.02 0.0438

Region               Red Hills Lake County       1 -7.5242 6.0122   -1.25 0.2108

Region               Redwood Valley       1 -12.7751 18.5626   -0.69 0.4913

Region               Sierra Foothills       1 -5.7533 5.0568   -1.14 0.2553

Region               Sonoma       1 0.6190 3.1954    0.19 0.8464

TextCluster\_SVD11                                1 14.8329 3.3378 4.44 <.0001

TextCluster\_SVD12                                1 40.9614 3.5060 11.68 <.0001

TextCluster\_SVD13                                1 18.0297 3.5191 5.12 <.0001

TextCluster\_SVD14                                1 9.0717 3.5913 2.53 0.0116

TextCluster\_SVD15                                1 9.9472 3.6038 2.76 0.0058

TextCluster\_SVD16                                1 -12.0296 3.6666 -3.28 0.0010

TextCluster\_SVD19                                1 7.9600 3.7081 2.15 0.0319

TextCluster\_SVD2                                 1 -10.7025 2.0866 -5.13 <.0001

TextCluster\_SVD20                                1 16.6686 3.7277 4.47 <.0001

TextCluster\_SVD21                                1 26.4049 3.7958 6.96 <.0001

TextCluster\_SVD23                                1 17.6138 3.8406 4.59 <.0001

TextCluster\_SVD24                                1 16.1919 3.7594 4.31 <.0001

TextCluster\_SVD30                                1 23.3678 3.8883 6.01 <.0001

TextCluster\_SVD32                                1 -17.8783 4.2180 -4.24 <.0001

TextCluster\_SVD39                                1 9.9056 4.2266 2.34 0.0191

TextCluster\_SVD4                                 1 -23.6416 2.8147 -8.40 <.0001

TextCluster\_SVD41                                1 -8.6404 4.1730 -2.07 0.0384

TextCluster\_SVD45                                1 9.0801 4.2532 2.13 0.0328

TextCluster\_SVD5                                 1 25.7882 2.9712 8.68 <.0001

TextCluster\_SVD6                                 1 -6.3382 3.1356 -2.02 0.0433

TextCluster\_SVD8                                 1 18.6213 3.4412 5.41 <.0001

TextCluster\_SVD9                                 1 13.4913 3.3761 4.00 <.0001

points                                           1 6.0189 0.1696 35.49 <.0001

**PYTHON**

import pandas as pd  
import string  
import nltk  
from nltk import pos\_tag  
from nltk.tokenize import word\_tokenize  
from nltk.stem.snowball import SnowballStemmer  
from nltk.stem import WordNetLemmatizer  
from nltk.corpus import wordnet as wn  
from nltk.corpus import stopwords  
from sklearn.feature\_extraction.text import CountVectorizer  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.feature\_extraction.text import TfidfTransformer  
from sklearn.decomposition import LatentDirichletAllocation  
  
#  
# nltk.download('punkt')  
# nltk.download('averaged\_perceptron\_tagger')  
# nltk.download('stopwords')  
# nltk.download('wordnet')  
  
  
# my\_analyzer replaces both the preprocessor and tokenizer  
# it also replaces stop word removal and ngram constructions  
  
def my\_analyzer(s):  
 # Synonym List  
 syns = {'veh': 'vehicle', 'car': 'vehicle', 'chev': 'cheverolet', \  
 'chevy': 'cheverolet', 'air bag': 'airbag', \  
 'seat belt': 'seatbelt', "n't": 'not', 'to30': 'to 30', \  
 'wont': 'would not', 'cant': 'can not', 'cannot': 'can not', \  
 'couldnt': 'could not', 'shouldnt': 'should not', \  
 'wouldnt': 'would not', }  
  
 # Preprocess String s  
 s = s.lower()  
 s = s.replace(',', '. ')  
 # Tokenize  
 tokens = word\_tokenize(s)  
 tokens = [word.replace(',', '') for word in tokens]  
 tokens = [word for word in tokens if ('\*' not in word) and \  
 ("''" != word) and ("``" != word) and \  
 (word != 'description') and (word != 'dtype') \  
 and (word != 'object') and (word != "'s")]  
  
 # Map synonyms  
 for i in range(len(tokens)):  
 if tokens[i] in syns:  
 tokens[i] = syns[tokens[i]]  
  
 # Remove stop words  
 punctuation = list(string.punctuation) + ['..', '...']  
 pronouns = ['i', 'he', 'she', 'it', 'him', 'they', 'we', 'us', 'them']  
 stop = stopwords.words('english') + punctuation + pronouns  
 filtered\_terms = [word for word in tokens if (word not in stop) and \  
 (len(word) > 1) and (not word.replace('.', '', 1).isnumeric()) \  
 and (not word.replace("'", '', 2).isnumeric())]  
  
 # Lemmatization & Stemming - Stemming with WordNet POS  
 # Since lemmatization requires POS need to set POS  
 tagged\_words = pos\_tag(filtered\_terms, lang='eng')  
 # Stemming with for terms without WordNet POS  
 stemmer = SnowballStemmer("english")  
 wn\_tags = {'N': wn.NOUN, 'J': wn.ADJ, 'V': wn.VERB, 'R': wn.ADV}  
 wnl = WordNetLemmatizer()  
 stemmed\_tokens = []  
 for tagged\_token in tagged\_words:  
 term = tagged\_token[0]  
 pos = tagged\_token[1]  
 pos = pos[0]  
 try:  
 pos = wn\_tags[pos]  
 stemmed\_tokens.append(wnl.lemmatize(term, pos=pos))  
 except:  
 stemmed\_tokens.append(stemmer.stem(term))  
 return stemmed\_tokens  
  
  
# Further Customization of Stopping and Stemming using NLTK  
def my\_preprocessor(s):  
 # Vectorizer sends one string at a time  
 s = s.lower()  
 s = s.replace(',', '. ')  
 print("preprocessor")  
 return (s)  
  
  
def my\_tokenizer(s):  
 # Tokenize  
 print("Tokenizer")  
 tokens = word\_tokenize(s)  
 tokens = [word.replace(',', '') for word in tokens]  
 tokens = [word for word in tokens if word.find('\*') != True and \  
 word != "''" and word != "``" and word != 'description' \  
 and word != 'dtype']  
 return tokens  
  
  
# Increase Pandas column width to let pandas read large text columns  
pd.set\_option('max\_colwidth', 32000)  
# Read GMC Ignition Recall Comments from NTHSA Data  
#file\_path = '/Users/Home/Desktop/python/Excel/'  
df = pd.read\_excel("wine.xlsx")  
  
# Setup simple constants  
n\_docs = len(df['description'])  
n\_samples = n\_docs  
m\_features = None  
s\_words = 'english'  
ngram = (1,2)  
  
# Setup reviews in list 'discussions'  
discussions = []  
for i in range(n\_samples):  
 discussions.append(("%s" %df['description'].iloc[i]))  
  
# Create Word Frequency by Review Matrix using Custom Analyzer  
cv = CountVectorizer(max\_df=0.95, min\_df=2, max\_features=m\_features,\  
 analyzer=my\_analyzer, ngram\_range=ngram)  
tf = cv.fit\_transform(discussions)  
  
print("\nVectorizer Parameters\n", cv, "\n")  
  
  
# LDA For Term Frequency x Doc Matrix  
n\_topics = 15  
max\_iter = 5  
learning\_offset = 20.  
learning\_method = 'online'  
# In sklearn, LDA is synonymous with SVD (according to their doc)  
lda = LatentDirichletAllocation(n\_components=n\_topics, max\_iter=max\_iter,\  
 learning\_method=learning\_method, \  
 learning\_offset=learning\_offset, \  
 random\_state=12345)  
lda.fit\_transform(tf)  
print('{:.<22s}{:>6d}'.format("Number of Reviews", tf.shape[0]))  
print('{:.<22s}{:>6d}'.format("Number of Terms", tf.shape[1]))  
print("\nTopics Identified using LDA")  
tf\_features = cv.get\_feature\_names()  
max\_words = 15  
for topic\_idx, topic in enumerate(lda.components\_):  
 message = "Topic #%d: " %topic\_idx  
 message += " ".join([tf\_features[i]  
 for i in topic.argsort()[:-max\_words - 1:-1]])  
 print(message)  
 print()  
  
# LDA for TF-IDF x Doc Matrix  
# First Create Term-Frequency/Inverse Doc Frequency by Review Matrix  
# This requires constructing Term Freq. x Doc. matrix first  
tf\_idf = TfidfTransformer()  
print("\nTF-IDF Parameters\n", tf\_idf.get\_params(),"\n")  
tf\_idf = tf\_idf.fit\_transform(tf)  
# Or you can construct the TF/IDF matrix from the data  
tfidf\_vect = TfidfVectorizer(max\_df=0.95, min\_df=2, max\_features=m\_features,\  
 analyzer=my\_analyzer, ngram\_range=ngram)  
tf\_idf = tfidf\_vect.fit\_transform(discussions)  
print("\nTF\_IDF Vectorizer Parameters\n", tfidf\_vect, "\n")  
  
lda = LatentDirichletAllocation(n\_components=n\_topics, max\_iter=max\_iter,\  
 learning\_method=learning\_method, \  
 learning\_offset=learning\_offset, \  
 random\_state=12345)  
lda.fit\_transform(tf\_idf)  
print('{:.<22s}{:>6d}'.format("Number of Reviews", tf.shape[0]))  
print('{:.<22s}{:>6d}'.format("Number of Terms", tf.shape[1]))  
print("\nTopics Identified using LDA with TF\_IDF")  
tf\_features = cv.get\_feature\_names()  
max\_words = 15  
for topic\_idx, topic in enumerate(lda.components\_):  
 message = "Topic #%d: " % topic\_idx  
 message += " ".join([tf\_features[i]  
 for i in topic.argsort()[:-max\_words - 1:-1]])  
 print(message)  
 print()

Vectorizer Parameters

CountVectorizer(analyzer=<function my\_analyzer at 0x100561e18>, binary=False,

decode\_error='strict', dtype=<class 'numpy.int64'>,

encoding='utf-8', input='content', lowercase=True, max\_df=0.95,

max\_features=None, min\_df=2, ngram\_range=(1, 2), preprocessor=None,

stop\_words=None, strip\_accents=None,

token\_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, vocabulary=None)

Number of Reviews..... 13135

Number of Terms....... 6263

Topics Identified using LDA

Topic #0: nose caramel palate vanilla rather bottle blueberry roast reserve peak aroma element bake atlas one-dimensional

Topic #1: interesting chewy tangy luxurious several raisins site seamlessly problem saddle burst marry eucalyptus super two

Topic #2: flavor blackberry cherry wine oak dry tannin soft drink finish black currant ripe fruit cabernet

Topic #3: green get thin ageability graphite sip pepper sweetly note star minty bouquet decent fruit glass

Topic #4: slightly may form within expect richly slight level textured layered herbal alcohol cool lushness wine

Topic #5: mountain wine fruit mark need vineyard time beyond come big tannin anoth together powerful satisfy

Topic #6: new mineral want vintage real cab french oak opulent sour great value price wine acidic

Topic #7: cocoa appeal power concentration bitter old density sizable tightly couple never blackcurrant herbaceous world wound

Topic #8: blackberry flavor cabernet currant year wine tannin dry oak rich black drink ripe cab show

Topic #9: solid sweetness case paso linger vanilla production core produce roble focus spice backbone intensely forest

Topic #10: black wine valley palate show dark fruit tannin vineyard cedar nose red cherry olive napa

Topic #11: fine bottle complexity year develop frame oakville great sonoma three reward pie beautiful next additional

Topic #12: especially opulence fleshy velvet firmly iron record consider track create special fist glove fat gracefully

Topic #13: like taste flavor sweet cherry blackberry wine alcohol soft seem fruit raisin almost hot little

Topic #14: wine cabernet merlot blend verdot tannin petit finish red black juicy franc oak sauvignon soft

TF-IDF Parameters

{'norm': 'l2', 'smooth\_idf': True, 'sublinear\_tf': False, 'use\_idf': True}

TF\_IDF Vectorizer Parameters

TfidfVectorizer(analyzer=<function my\_analyzer at 0x100561e18>, binary=False,

decode\_error='strict', dtype=<class 'numpy.int64'>,

encoding='utf-8', input='content', lowercase=True, max\_df=0.95,

max\_features=None, min\_df=2, ngram\_range=(1, 2), norm='l2',

preprocessor=None, smooth\_idf=True, stop\_words=None,

strip\_accents=None, sublinear\_tf=False,

token\_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use\_idf=True,

vocabulary=None)

Number of Reviews..... 13135

Number of Terms....... 6263

Topics Identified using LDA with TF\_IDF

Topic #0: muscular small-production breadth asian red-cherry mixed longtime mountain-grown penetrate crowd-pleasing meatiness michael lip-smacking beaulieu float

Topic #1: richer orange section zest rosé alongsid toughly brushy roundness olallieberry cloves elongate slow program verging

Topic #2: flavor blackberry wine cherry dry currant cabernet oak tannin drink ripe year sweet rich cab

Topic #3: bean join farm sultry distinctive black-olive western crack affordably fruit-driven plushness neighbor suggestive restrained boisterous

Topic #4: rusticity dustiness astringently unevenly thread multiple cabernet-like tannin-acid concord handsome ginger stubborn cloud 2023–2033 punchy

Topic #5: porty barely harsh eucalyptus acceptable compost cough dot vegetal heavily likable delight heavy-handed echo tiny-production

Topic #6: tad flat showcasing damp cake bay memorable spent abundance disjoint shin grapy greet philippe tread

Topic #7: george beckstoffer vibrancy impart umami rhubarb iii stemmy midway six-plus stretch risk ahead red-fruit quaff

Topic #8: today opposite lightness luxuriously terribly gooey seduces separate exploration belies medium-length 2009–2015 praiseworthy loud backbon

Topic #9: pairing own pliant ferment oak-like oregano zin reviewer william sea state foley mustiness slide condense

Topic #10: light-bodied graceful somehow quiet recall black-plum capture abundant softens aging darkly unfurl la stewy urge

Topic #11: lend gamy conveys sport high-elevation nick goldschmidt weedy drinker land tightness akin indicate intertwine vanilla-tinged

Topic #12: sweaty refresh pillowy unctuous brother funky celebration chile status deserves withstand similarly positive sister daou

Topic #13: cardamom overshadow black-fruit associate nickel tamp son baldacci masculine company disturb phelps whatev tingle vision

Topic #14: wine black tannin finish fruit palate flavor cabernet red cherry oak cedar soft dry blackberry

Process finished with exit code 0