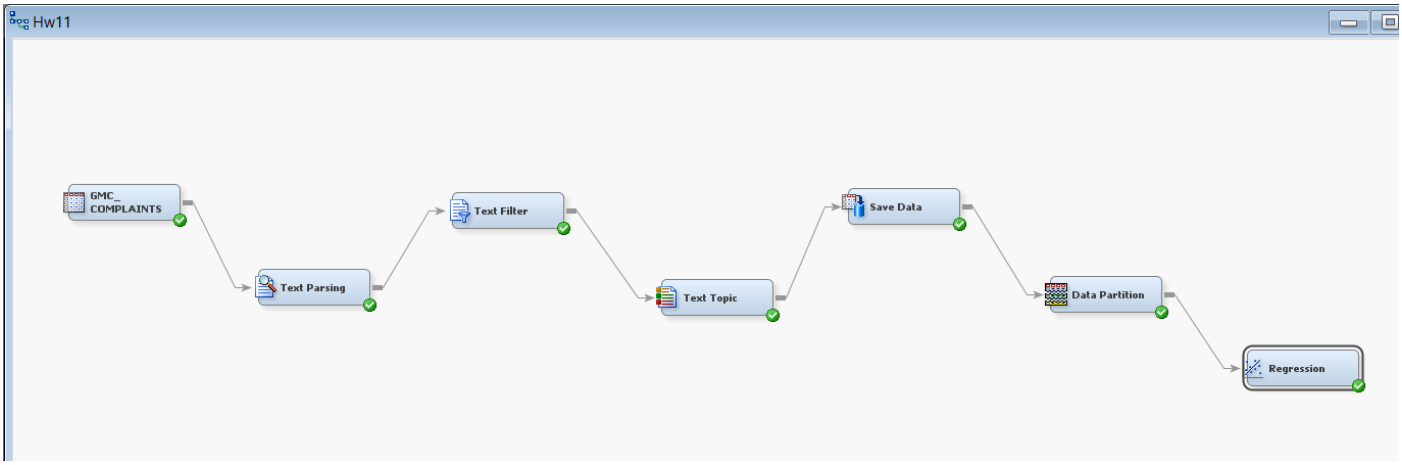
# STAT 656
# Week 11 Assignment

Name-Mayank Jaggi

UIN-526005299

# PART 1 SAS EM

## 1) Project Diagram & Property Window




Text Parsing Property

| .. Property | Value |
|---|---|
| **General** | |
| Node ID | TextParsing |
| Imported Data | |
| Exported Data | |
| Notes | |
| **Train** | |
| Variables | |
| Parse | |
| Parse Variable | Description |
| Language | English |
| Detect | |
| Different Parts of Speech | Yes |
| Noun Groups | Yes |
| Multi-word Terms | SASHELP.ENG_MULTI |
| Find Entities | None |
| Custom Entities | |
| Ignore | |
| Ignore Parts of Speech | 'Aux' 'Conj' 'Det' 'Interj' 'Part' 'Prep' 'Pr |
| Ignore Types of Entities | |
| Ignore Types of Attributes | 'Num' 'Punct' |
| Synonyms | |
| Stem Terms | Yes |
| Synonyms | SASHELP.ENGSYNMS |
| Filter | |
| Start List | |
| Stop List | SASHELP.ENGSTOP |
| Select Languages | |
| **Report** | |
| Number of Terms to Display | 20000 |
| **Status** | |
| Create Time | 4/10/19 1:11 PM |
| Run ID | 680b8b80-9570-45b7-a14c-fe48c1e8887 |
| Last Error | |
| Last Status | Complete |
| Last Run Time | 4/10/19 1:15 PM |
| Run Duration | 0 Hr. 0 Min. 14.32 Sec. |
| Grid Host | |
| User-Added Node | No |

## Text Filter Property

| .. Property | Value |
|---|---|
| **General** | |
| Node ID | TextFilter |
| Imported Data | |
| Exported Data | |
| Notes | |
| **Train** | |
| Variables | |
| Spelling | |
| Check Spelling | No |
| Dictionary | |
| Weightings | |
| Frequency Weighting | None |
| Term Weight | Inverse Document Frequency |
| Term Filters | |
| Minimum Number of Documents | 10 |
| Maximum Number of Terms | . |
| Import Synonyms | |
| Document Filters | |
| Search Expression | |
| Subset Documents | |
| Results | |
| Filter Viewer | |
| Spell-Checking Results | |
| Exported Synonyms | |
| **Report** | |
| Terms to View | All |
| Number of Terms to Display | 20000 |
| **Status** | |
| Create Time | 4/10/19 1:12 PM |
| Run ID | dd51f76a-4b2e-42f1-8d1f-9d4fa4005c41 |
| Last Error | |
| Last Status | Complete |
| Last Run Time | 4/10/19 1:21 PM |
| Run Duration | 0 Hr. 0 Min. 3.99 Sec. |
| Grid Host | |
| User-Added Node | No |

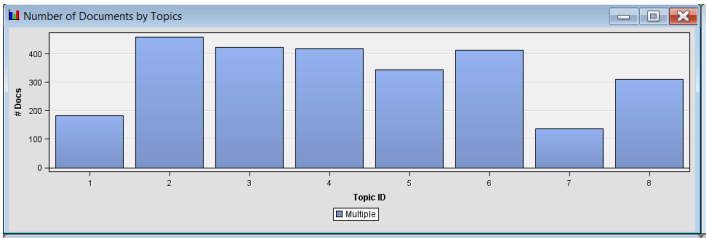
## Text Topic Property

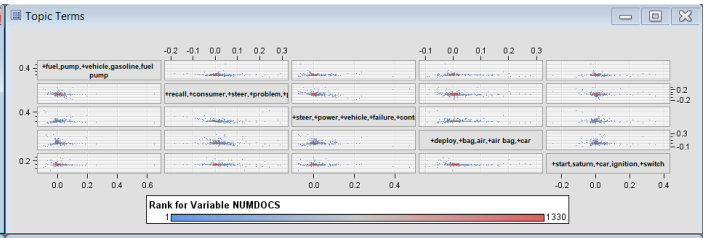
| .. Property | Value |
|---|---|
| **General** | |
| Node ID | TextTopic |
| Imported Data | |
| Exported Data | |
| Notes | |
| **Train** | |
| Variables | |
| User Topics | |
| Term Topics | |
| Number of Single-term Topics | 0 |
| Learned Topics | |
| Number of Multi-term Topics | 8 |
| Correlated Topics | No |
| Results | |
| Topic Viewer | |
| **Status** | |
| Create Time | 4/10/19 1:12 PM |
| Run ID | 0bdf209b-4623-4842-bc15-fd30a91d3ae |
| Last Error | |
| Last Status | Complete |
| Last Run Time | 4/10/19 1:23 PM |
| Run Duration | 0 Hr. 0 Min. 4.46 Sec. |
| Grid Host | |
| User-Added Node | No |

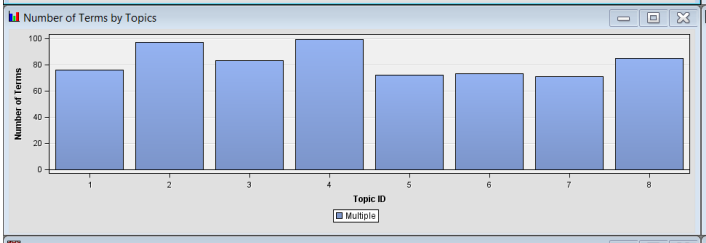
# Regression Property

| Property | Value |
|---|---|
| **General** | |
| Node ID | Reg |
| Imported Data | |
| Exported Data | |
| Notes | |
| **Train** | |
| Variables | |
| Equation | |
| Main Effects | Yes |
| Two-Factor Interactions | No |
| Polynomial Terms | No |
| Polynomial Degree | 2 |
| User Terms | No |
| Term Editor | |
| Class Targets | |
| Regression Type | Logistic Regression |
| Link Function | Logit |
| Model Options | |
| Suppress Intercept | No |
| Input Coding | Deviation |
| Model Selection | |
| Selection Model | Stepwise |
| Selection Criterion | Default |
| Use Selection Defaults | Yes |
| Selection Options | |
| Optimization Options | |
| Technique | Default |
| Default Optimization | Yes |
| Max Iterations | 0 |
| Max Function Calls | 0 |
| Maximum Time | 1 Hour |
| Convergence Criteria | |
| Uses Defaults | Yes |
| Options | |
| Output Options | |
| Confidence Limits | No |
| Save Covariance | No |
| Covariance | No |
| Correlation | No |
| Statistics | No |
| Suppress Output | No |
| Details | No |
| Design Matrix | No |
| **Score** | |
| Excluded Variables | Reject |
| **Status** | |
| Create Time | 4/10/19 1:14 PM |
| Run ID | 9ff5887a-7ff7-4ab9-8ae4-75a324813f8 |
| Last Error | |
| Last Status | Complete |
| Last Run Time | 4/10/19 1:29 PM |

**2)**




Terms

| Term | Role | Attribute | WEIGHT | Freq | # Docs | Keep | Rank for Variable NUMDOCS | +fuel,pump,+vehicle,gasoline,fuel pump | +recall,+consumer,+steer,+problem,+power | +steer,+power,+vehicle,+failure,+contact | +deploy,+bag,air,+air bag,+car | + |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| + vehicle ... | Noun | Alpha | 1.923889 | 3582 | 1440 | Y | 1 | 0.194 | -0.049 | 0.237 | 0.181 | |
| + steer ... | Verb | Alpha | 2.023424 | 3173 | 1344 | Y | 2 | -0.093 | 0.244 | 0.47 | -0.058 | |
| tr ... | Prop | Alpha | 2.036363 | 1333 | 1332 | Y | 3 | 0.005 | 0.103 | 0.046 | 0.032 | |
| + car ... | Noun | Alpha | 2.12227 | 3564 | 1255 | Y | 4 | 0.085 | 0.123 | -0.067 | 0.203 | |
| + power ... | Noun | Alpha | 2.340263 | 2081 | 1079 | Y | 5 | -0.079 | 0.215 | 0.388 | -0.086 | |
| + failure ... | Noun | Alpha | 2.496879 | 1747 | 968 | Y | 6 | 0.06 | -0.144 | 0.225 | 0.004 | |
| + cobalt ... | Noun | Alpha | 2.514875 | 1143 | 956 | Y | 7 | 0.031 | 0.139 | 0.072 | 0.042 | |
| + contact ... | Noun | Alpha | 2.689524 | 2285 | 847 | Y | 8 | 0.098 | -0.182 | 0.209 | 0.003 | |

Topics

| Category | Topic ID | Document Cutoff | Term Cutoff | Topic | Number of Terms | # Docs |
|---|---|---|---|---|---|---|
| Multiple | 1 | 0.173 | 0.033 | +fuel,pump,+vehic... | 76 | 182 |
| Multiple | 2 | 0.177 | 0.036 | +recall,+consume... | 97 | 460 |
| Multiple | 3 | 0.269 | 0.034 | +steer,+power,+v... | 83 | 423 |
| Multiple | 4 | 0.168 | 0.036 | +deploy,+bag,air,... | 99 | 419 |
| Multiple | 5 | 0.161 | 0.035 | +start,saturn,+car... | 72 | 344 |
| Multiple | 6 | 0.224 | 0.033 | +contact,campaig... | 73 | 414 |
| Multiple | 7 | 0.155 | 0.034 | +consumer,ignitio... | 71 | 136 |
| Multiple | 8 | 0.143 | 0.035 | +door,+key,ignitio... | 85 | 310 |

Output

```
1  *------------------------------------------------------------*
2  User:                  mayank
3  Date:                  April 10, 2019
4  Time:                  13:29:22
5  *------------------------------------------------------------*
6  * Training Output
7  *------------------------------------------------------------*
8
9
10
11
12 Variable Summary
13
14                Measurement    Frequency
15  Role             Level          Count
16
```

### 3) Confusion Matrix

| | | Predicted | |
|---|---|---|---|
| | | Negative | Positive |
| Actual | Negative | 582 | 19 |
| | Positive | 93 | 127 |

### 4) Results

| | |
|---|---|
| Accuracy | 0.863581 |
| Precision | 0.869863 |
| Recall | 0.577273 |
| F1 | 0.693989 |

PART 2 PYTHON

1) PYTHON CODE

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 10 10:41:13 2019

@author: mayank
"""
import pandas as pd
import string
import nltk
import numpy as np
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

#For regression
from AdvancedAnalytics import ReplaceImputeEncode
from AdvancedAnalytics import logreg
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

#Analyzer
def 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', 'air':'airbag', 'bag':'airbag'}


    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']
    # For adding extra stop words
    other = ['own','go','get','seem','say','would','regard','report','involve'\
             ,'do','anoth','consumer',"'ve",'happen','try','either','come',]
    stop = stopwords.words('english') + punctuation + pronouns + other
    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())]

    # Stemming
    tagged_words = pos_tag(filtered_terms, lang='eng')
    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

# Stopping and Stemming using NLTK
def preprocessor(s):
    #Vectorizer sends one string at a time
    s = s.lower()
    s = s.replace(',', '. ')
    print("preprocessor")
    return(s)

def 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
pd.set_option('max_colwidth', 32000)
# California Cabernet Reviews
df = pd.read_excel("GMC_Complaints.xlsx")

# Setup simple constants
n_docs     = len(df['description'])
n_samples  = n_docs
m_features = 1000
s_words    = 'english'
ngram = (1,2)
max_df=0.8

discussions = []
for i in range(n_samples):
    discussions.append(("%s" %df['description'].iloc[i]))


# Create Word Frequency
cv = CountVectorizer(max_df=max_df, min_df=2, max_features=m_features,\
                     analyzer=analyzer, ngram_range=ngram)
tf = cv.fit_transform(discussions)

n_topics        = 8
max_iter        =  5
learning_offset = 20.
learning_method = 'online'

tf_idf = TfidfTransformer()
print("\nTF-IDF Parameters\n", tf_idf.get_params(),"\n")
tf_idf = tf_idf.fit_transform(tf)

tfidf_vect = TfidfVectorizer(max_df=max_df, min_df=2, max_features=m_features,\
                             analyzer=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
desc = []
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()
        desc.append([tf_features[i] for i in topic.argsort()[:-max_words - 1:-1]])

#Extract topic probablities
topics = pd.DataFrame(lda.fit_transform(tf_idf))
preds = ['Year', 'make','model','crashed','abs','mileage']
reg_df = pd.concat([df[preds],topics], axis=1, ignore_index=True)
reg_df.columns = ['Year',
'make','model','crashed','abs','mileage','0','1','2','3','4','5','6','7']

# logistics regression model

attribute_map = {
        'Year'    :['I',(2003,2011),[0,0]],
        'make'    :['N',('CHEVROLET','PONTIAC','SATURN'),[0,0]],
        'model'   :['N',('COBALT','G5','HHR','ION','SKY','SOLSTICE'),[0,0]],
        'crashed' :['B',('N','Y'),[0,0]],
        'abs'     :['B',('N','Y'),[0,0]],
        'mileage' :['I',(0,200000),[0,0]],
        '0'       :['I',(0,1),[0,0]],
        '1'       :['I',(0,1),[0,0]],
        '2'       :['I',(0,1),[0,0]],
        '3'       :['I',(0,1),[0,0]],
        '4'       :['I',(0,1),[0,0]],
        '5'       :['I',(0,1),[0,0]],
        '6'       :['I',(0,1),[0,0]],
        '7'       :['I',(0,1),[0,0]],
}
varlist = ['crashed']
rie = ReplaceImputeEncode(data_map=attribute_map, \
                                nominal_encoding='one-hot',
                           interval_scale = None, drop=False, display=False)
encoded_df = rie.fit_transform(reg_df)
X = encoded_df.drop(varlist, axis=1)
y = encoded_df[varlist]
np_y=np.ravel(y)

max_f1 = 0
C_list=[.1,1,10,100]
score_list = ['accuracy', 'recall', 'precision', 'f1']
for c in C_list:
    print("\nRegularization Parameter: ", c)
```

```
    lgr = LogisticRegression(C=c, tol=1e-8, max_iter=1000)
    lgr.fit(X, np_y)
    scores = cross_validate(lgr, X, np_y,\
                            scoring=score_list, return_train_score=False, \
                            cv=10)
    print("{:.<13s}{:>6s}{:>13s}".format("Metric", "Mean", "Std. Dev."))
    for s in score_list:
        var = "test_"+s
        mean = scores[var].mean()
        std  = scores[var].std()
        print("{:.<13s}{:>7.4f}{:>10.4f}".format(s, mean, std))
        if mean > max_f1:
            max_f1 = mean
            best_predictor  = c
print("\nBest based on F1-Score")
print("Best Regularization Parameter = ", best_predictor)

X_train, X_valid, y_train, y_valid= \
train_test_split(X,y,test_size = 0.3, random_state=7)

np_y_train = np.ravel(y_train)
np_y_valid = np.ravel(y_valid)

reg = LogisticRegression(C=best_predictor, tol=1e-8, max_iter=1000)
reg.fit(X_train,np_y_train)

logreg.display_coef(reg,21,2,X_train.columns)

logreg.display_binary_split_metrics(reg,X_train,np_y_train,X_valid,np_y_valid)
```

```
##########################################################################################
##########
OUTPUT

2) MODEL METRICS

Model Metrics.........       Training     Validation
Observations..........           1913            821
Coefficients..........             21             21
DF Error..............           1892            800
Mean Absolute Error....         0.2712         0.2967
Avg Squared Error......         0.1317         0.1516
Accuracy..............          0.8191         0.7881
Precision.............          0.8818         0.8554
Recall (Sensitivity)...         0.3573         0.3047
F1-score..............          0.5085         0.4494
MISC (Misclassification)...      18.1%          21.2%
```

| | | |
|---|---|---|
| class 0..... | 1.7% | 2.0% |
| class 1..... | 64.3% | 69.5% |

3) CONFUSION MATRIX

Training

| Confusion Matrix | Class 0 | Class 1 |
|---|---|---|
| Class 0..... | 1388 | 24 |
| Class 1..... | 322 | 179 |

Validation

| Confusion Matrix | Class 0 | Class 1 |
|---|---|---|
| Class 0..... | 576 | 12 |
| Class 1..... | 162 | 71 |