CIS 4130

Amanjit Singh

[AMANJIT.SINGH1@baruchmail.cuny.edu](mailto:AMANJIT.SINGH1@baruchmail.cuny.edu)

Proposal:

I plan on using the Amazon Customer Reviews dataset. There is data that spans over 2 decades, starting from 1995. Within this data set there is a lot of information one can use to understand customer product experiences. This dataset in particular is a representative sample of opinions, different views of products across the world, with promotional/bias reviews. The link to this dataset is: <https://www.kaggle.com/datasets/cynthiarempel/amazon-us-customer-reviews-dataset>. It is from Kaggle, and after applying a 10gb minimum filter, you can find this dataset on the second page of results. The attributes in this dataset are marketplace, customer\_id, review\_id, product\_id, product\_parent, product\_title, product\_category, star\_rating, helpful\_votes, total\_votes, vine, verified\_purchase, review\_headline, review\_body, and review\_date. The vine attributes are reviews that were written as a part of the vine program. With this data, I would like to see how people react to different products and be able to give to a company the most optimal way of creating their next product. For example, the machine learning algorithm used for region 1 using the product knives finds that customers like it when the knives have a rubber handle and hate the wooden handle. Additionally, more people are willing to buy the knife if it comes with a blade cover and a edge sharpener. With this information, I would be able to maybe predict future reviews and the number of stars the of the optimal product and then go to a knife manufacturer that works in region 1 and be able to give them all of this information to increase their sales.

Milestone #2:

To start the data acquisition process, I used my admin user account to create an linux AMI instance that would allow me to also use the AWS CLI. I then downloaded python and Kaggle to be able to use its CLI and then I entered the credentials to be able to view the datasets and be able to download them. After listing the dataset, I was using, I started downloading the files one at a time. After downloading them, I unzip the .zip files and move them to the landing folder in my s3 bucket. Due to some issues of Kaggle, there are some files I wasn’t able to download directly. To resolve this issue, I downloaded all the data at once and then unziped, moved, and removed the files that I didn’t already have one at a time one at a time. After adding all of the missing files, I removed the .zip kaggle database that I downloaded.

**Code:**

kaggle datasets download -d cynthiarempel/amazon-us-customer-reviews-dataset

unzip amazon-us-customer-reviews-dataset.zip amazon\_reviews\_us\_Outdoors\_v1\_00.tsv

unzip amazon-us-customer-reviews-dataset.zip amazon\_reviews\_us\_PC\_v1\_00.tsv

unzip amazon-us-customer-reviews-dataset.zip amazon\_reviews\_us\_Personal\_Care\_Appliances\_v1\_00.tsv

aws s3 cp amazon\_reviews\_us\_Outdoors\_v1\_00.tsv s3://4130semesterprojectas/landing/ amazon\_reviews\_us\_Outdoors\_v1\_00.tsv

aws s3 cp amazon\_reviews\_us\_PC\_v1\_00.tsv s3://4130semesterprojectas/landing/ amazon\_reviews\_us\_PC\_v1\_00.tsv

aws s3 cp amazon\_reviews\_us\_Personal\_Care\_Appliances\_v1\_00.tsv s3://4130semesterprojectas/landing/ amazon\_reviews\_us\_Personal\_Care\_Appliances\_v1\_00.tsv

rm amazon\_reviews\_us\_Outdoors\_v1\_00.tsv

rm amazon\_reviews\_us\_PC\_v1\_00.tsv

rm amazon\_reviews\_us\_Personal\_Care\_Appliances\_v1\_00.tsv

rm amazon-us-customer-reviews-dataset.zip

Milestone 3.

majorappliancedf -

A screenshot of a computer

Description automatically generatedA close up of a document

Description automatically generated

A graph of a number of blue bars

Description automatically generated

mobileelecdf -

A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated

A graph of a person

Description automatically generated

To work with this data, I would start with getting rid of the nulls. There are very few in the datasets, so it would be better to just take them out. I would also need to figure out how to filter out the documents. There is a column called verified purchase. I plan on filtering the reviews based on those of verified purchases. I also am thinking of filtering the reviews based on word count, however I am not sure if its possible. There can be 2-word reviews such as “very good”, which can be a pretty valuable piece of information no matter how simple it is, or it can be a 3-word review that adds no meaningful value, but that’s just how data is.

Milestone 4:

For this milestone I used data bricks to get the job done. I read in the csv file from my s3 bucket in aws, used the infer schema parameter to auto create the schema. I looked over the types of data to double check the infer schema was accurate. Then I selected the columns that I thought could be useful to filter out data and get more accurate results. I also saved the new copy of the data as a parquet file in the raw/ folder. After that I read in the files from the raw/ folder and started feature engineering. I started off by combining the review headline and body so I can use both text data in the model. I also tokenized the data using regex tokenizer so that regular expressions could be preserved. After that I used hashingtf and idf to turn the words into features. I also used textblob to find the sentiment score. After that I used binarizer to turn star ratings into binary ratings to pass into the logistic regression model. To save time, I only limited the data to the top 20 rows when working with the code. This is because trying to use the whole data takes at least 20 minutes and not even 1 job was completed. Having to wait for the code to finish executing is inconvenient. Since I split the data 80/20 with 20 rows, there are only 4 points being tested. The accuracy of the model was 40%, and precision was 50%, recall was 66% and the f1 score was 57%.

Milestone 5:

A bar code with text

Description automatically generated

This plot shows the frequency of helpful votes within my dataset. We can see that most of the helpful votes are under 50, and after 50 the bars start to separate more.

A graph with blue dots and numbers

Description automatically generated

This plot shows the relationship between star ratings and sentiment score. We can see a general positive correlation even though there are some outliers.

A computer code with numbers and symbols

Description automatically generated

This shows the performance metrics of the testing data. It would show the users how accurate and precises the model is. It also shows the recall and f1\_score to show me how reliable the model preforms.

A graph with blue bars

Description automatically generated

This graph shows the distribution of star ratings in the training dataset.

A green and white squares with white text

Description automatically generated

This heatmap shows the correlation between sentiment score, star rating, helpful votes, and binary ratings to each other. I am a bit surprised that helpful votes had such low correlation.

Milestone 6:

Starting with getting the Amazon reviews data from Kaggle, I created an API and used an AWS EC2 instance to download the dataset from the cloud and save it directly into my landing folder in the s3 bucket. I then used data bricks to load in a piece of the dataset and perform EDA. After setting up the schema, getting rid of unwanted columns, and printing descriptive sats of some columns, I saved it as a parquet file in my raw folder in the s3 bucket. From there, I loaded in the parquet file, performed feature engineering, and applied my model to the testing data. To check the model performance and data, I created visualizations of the model and checked the accuracy, precision, recall, and F1 score. From there I saved the data into my trusted folder and the models into the model folder.

Milestone 7:

GitHub URL: https://github.com/wllothewisp/classcoding.git

Appendix:

A screenshot of a computer

Description automatically generated

A close up of a computer screen

Description automatically generated

A screen shot of a computer code

Description automatically generatedA computer screen shot of a code

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated A computer screen shot of a program code

Description automatically generated

A screenshot of a computer code

Description automatically generated

Milestone 4:

Reading 1st file:

file\_path = 's3a://4130semesterprojectas/landing/amazon\_reviews\_us\_Apparel\_v1\_00.tsv'

sdf = spark.read.csv(file\_path, sep='\t', header=True, inferSchema=True)

from pyspark.sql.functions import col, input\_file\_name

cleaned\_data = (

    sdf

    .select(

        col("marketplace"),

        col("review\_id"),

        col("product\_id"),

        col("product\_parent"),

        col("product\_title"),

        col("product\_category"),

        col("star\_rating"),

        col("helpful\_votes")

    )

    .na.drop()

)

s3\_bucket = "s3://4130semesterprojectas/raw/"

cleaned\_data.write.mode("append").parquet(s3\_bucket)

Reading 2nd file:

file\_path = 's3a://4130semesterprojectas/landing/amazon\_reviews\_us\_Shoes\_v1\_00.tsv'

sdf = spark.read.csv(file\_path, sep='\t', header=True, inferSchema=True)

cleaned\_data = (

    sdf

    .select(

        col("marketplace"),

        col("review\_id"),

        col("product\_id"),

        col("product\_parent"),

        col("product\_title"),

        col("product\_category"),

        col("star\_rating"),

        col("helpful\_votes"),

        col("verified\_purchase"),

        col("review\_headline"),

        col("review\_body")

    )

    .na.drop()

)

s3\_bucket = "s3://4130semesterprojectas/raw/"

cleaned\_data.write.mode("append").parquet(s3\_bucket)

FE and Model :

from pyspark.ml.feature import RegexTokenizer, HashingTF, IDF

from pyspark.sql.functions import concat, col

from textblob import TextBlob

from pyspark.sql.types import DoubleType

from pyspark.sql.functions import col, isnan, when, count, udf

sc.setLogLevel("ERROR")

from pyspark.sql.functions import \*

from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler

from pyspark.ml import Pipeline

from pyspark.ml.classification import LogisticRegression

from pyspark.ml.evaluation import BinaryClassificationEvaluator

from pyspark.ml.tuning import CrossValidator, ParamGridBuilder

import numpy as np

import io

import s3fs

import seaborn as sns

import matplotlib.pyplot as plt

from IPython.display import display, Image

import pandas as pd

from pyspark.ml.stat import Correlation

from pyspark.ml.feature import Binarizer

file\_pattern = 's3a://4130semesterprojectas/raw/\*.parquet'

#sdf = spark.read.csv(file\_path, sep='\t', header=True, inferSchema=True)

rdf = spark.read.parquet(file\_pattern)

rdf = rdf[(rdf['verified\_purchase'] == 'Y') & (rdf['helpful\_votes'] > '1')]

columns\_to\_drop = ["marketplace", "product\_parent", "product\_title"]

rdf = rdf.drop(\*columns\_to\_drop)

rdf = rdf[(rdf['verified\_purchase'] == 'Y') & (rdf['helpful\_votes'] > '1')]

rdf = rdf.withColumn("full\_review", concat(col("review\_headline"), col("review\_body")))

rdf = rdf.withColumn("star\_rating", col("star\_rating").cast("double"))

regexTokenizer = RegexTokenizer(inputCol="full\_review", outputCol="words", pattern="\\w+", gaps=False)

hasher = HashingTF(numFeatures=4096, inputCol="words", outputCol="word\_features")

idf = IDF(inputCol='word\_features', outputCol="idffeatures", minDocFreq=1)

binarizer = Binarizer(threshold=3.0, inputCol="star\_rating", outputCol="binary\_rating")

feature\_col = "idffeatures"

label\_col = "binary\_rating"

lr = LogisticRegression(labelCol= label\_col, featuresCol= feature\_col)

pipeline = Pipeline(stages=[regexTokenizer, hasher, idf, binarizer])

rdf = pipeline.fit(rdf).transform(rdf)

def sentiment\_analysis(some\_text):

    sentiment = TextBlob(some\_text).sentiment.polarity

    return sentiment

sentiment\_analysis\_udf = udf(sentiment\_analysis, DoubleType())

rdf = rdf.withColumn("sentiment\_score",

 sentiment\_analysis\_udf( rdf['full\_review'] ))

train\_data, test\_data = rdf.randomSplit([0.8, 0.2], seed=42)

model = lr.fit(train\_data)

test\_results = model.transform(test\_data)

s3\_bucket = "s3://4130semesterprojectas/trusted/"

test\_results.write.mode("overwrite").parquet(s3\_bucket)

cm = test\_results.groupby('binary\_rating').pivot('prediction').count().fillna(0).collect()

def calculate\_recall\_precision(cm):

 tn = cm[0][1] # True Negative

 fp = cm[0][2] # False Positive

 fn = cm[1][1] # False Negative

 tp = cm[1][2] # True Positive

 precision = tp / ( tp + fp )

 recall = tp / ( tp + fn )

 accuracy = ( tp + tn ) / ( tp + tn + fp + fn )

 f1\_score = 2 \* ( ( precision \* recall ) / ( precision + recall ) )

 return accuracy, precision, recall, f1\_score

print( calculate\_recall\_precision(cm) )

Milestone #5:

helpful\_votes\_df = test\_results.groupby('helpful\_votes').count().sort('helpful\_votes').toPandas()

values = helpful\_votes\_df['helpful\_votes']

plt.hist(values, bins=1000, color='green', edgecolor='black')

plt.xlim(0, 200)

plt.title("Helpful Votes Freq")

img\_data = io.BytesIO()

plt.savefig(img\_data, format='png', bbox\_inches='tight')

img\_data.seek(0)

s3 = s3fs.S3FileSystem(anon=False)

with s3.open('s3://4130semesterprojectas/model/freq.png', 'wb') as f:

 f.write(img\_data.getbuffer())

df = test\_results.select('star\_rating', 'sentiment\_score').toPandas()

sns.set\_style("white")

lmp = sns.lmplot(x='star\_rating', y='sentiment\_score', data=df)

img\_data = io.BytesIO()

lmp.savefig(img\_data, format='png', bbox\_inches='tight')

img\_data.seek(0)

s3 = s3fs.S3FileSystem(anon=False)

with s3.open('s3://4130semesterprojectas/model/lmplot.png', 'wb') as f:

 f.write(img\_data.getbuffer())

df1 = test\_results.select('star\_rating').toPandas()

sns.set\_style("white")

distribution\_plot = sns.displot(df1)

img\_data = io.BytesIO()

distribution\_plot.savefig(img\_data, format='png', bbox\_inches='tight')

img\_data.seek(0)

s3 = s3fs.S3FileSystem(anon=False)

with s3.open('s3://4130semesterprojectas/model/distribution\_plot.png', 'wb') as f:

 f.write(img\_data.getbuffer())

vector\_column = "correlation\_features"

numeric\_columns  = ["sentiment\_score", "star\_rating", "helpful\_votes", "binary\_rating"]

assembler = VectorAssembler(inputCols=numeric\_columns, outputCol=vector\_column)

sdf\_vector = assembler.transform(rdf).select(vector\_column)

matrix = Correlation.corr(sdf\_vector, vector\_column).collect()[0][0]

correlation\_matrix = matrix.toArray().tolist()

correlation\_matrix\_df = pd.DataFrame(data=correlation\_matrix, columns=numeric\_columns,index=numeric\_columns)

sns.set\_style("white")

fig, ax = plt.subplots(figsize = (16,8))

hm = sns.heatmap(correlation\_matrix\_df, xticklabels=correlation\_matrix\_df.columns.values, yticklabels=correlation\_matrix\_df.columns.values, cmap="Greens", annot=True, ax=ax)

plt.title('Correlation Matrix')

figure = hm.get\_figure()

img\_data = io.BytesIO()

figure.savefig(img\_data, format='png', bbox\_inches='tight')

img\_data.seek(0)

s3 = s3fs.S3FileSystem(anon=False)

with s3.open('s3://4130semesterprojectas/model/correlation\_plot.png', 'wb') as f:

   f.write(img\_data.getbuffer())

display(Image(data=img\_data.getvalue()))