CIS 4130 Term Project

Ching Hsuan, Lin

CIS 4130

[Chinghsuan.lin@baruchmail.cuny.edu](mailto:Chinghsuan.lin@baruchmail.cuny.edu)

# Milestone 1: project proposal

The dataset I choose is Amazon US Customer Review dataset. In this dataset it has several files depending on their product categories. In the each table of a category, it has columns like product\_id , customer\_id, review\_id, product\_category, star\_rating, review\_headline, and review\_body. For this project, I think I can used product review to do the sentiment analysis from natural language processing techniques to divide the reviews by positive, neutral, and negative.

# Milestone 2: Data Acquistion

I transfer my dataset directly from Kaggle. I firstly set up my Command Line Interface in my EC2 instance. And I create a S3 bucket called mybucket4130. After that, I install the Kaggle command line interface. Then I copy my Kaggle API Token in the clipboard, and in the CLI I set up a Kaggle directory, create a new json file via nano editor, and paste my Kaggle API token then save and secure the file. One of the important steps is to edit two lines of kaggle\_api\_extended.py by using nano editor to accommodate writing the file to standard output. At the last step, I used the *Kaggle datasets list -s [KEYWORD]* to find my dataset then used *Kaggle datasets download -d [DATASET]* to download the dataset then append this line *aws s3 cp - s3://mybucket4130/amazonreview.zip* to directly transfer the dataset to my S3 bucket.

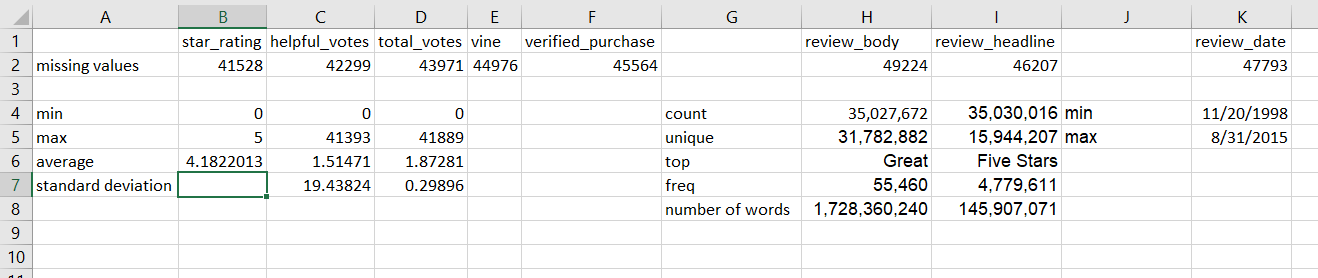
Graphical user interface, application

Description automatically generated

# Milestone 3: Exploratory Data Analysis

The dataset has around 35 million rows of data. The dataset I used have 15 columns, includes [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,review\_date] There are 5 numeric variables and 3 of them are id and the rest are helpful votes and total votes. I think the problem with cleaning and doing feature engineering would be some missing value, maybe some rows include some inconsistent data type, and maybe run out of memory would be another main problem when I do the feature engineering.

Statistics of data: (some columns are ID so there is no need to get the statistics about them)



# Milestone 4: Coding and Modeling

For this Amazon customer review dataset, I’m going to do sentiment analysis from Amazon customer that I want to classify the attitude through customers’ words in the reviews to see whether they are positive or negative to the products. In the beginning I would like to check all null values drop them since they are only small amount of data. The feature would be the customer reviews (review\_body), which I would need to do feature engineering to turn our raw review body into clean and without emoji and encode the customer rating to binary variable (positive, negative) and named the column “rating\_converted”.

def ascii\_only(mystring):

if mystring:

return mystring.encode('ascii', 'ignore').decode('ascii')

else:

return None

df = df.withColumn("rating\_converted", when(col("star\_rating") > 3, 'positive').otherwise('negative'))

I also create the machine learning pipeline which included tokenization, which converted sentence to into individual words. StopWordsRemover, which reomve the stopwords such as “the”, “an”, and “at” to remove lower level information and let model to focus on important words. HashingTF, it can convert words into vectors format. And with IDF can extract the feature from the results of HashingTF, which provide us word-relevance and can recognize how important within the documents. StringIndexer will help us to encode the rate to categorical variable such as positive become 0 and negative become 1. The last stage was the created the logistic regression. The reason I using this model because this regression was properly to be used when the dependent variable (label) has binary values. Besides, the results I’m going to predict is about customers’ emotion from their review to know whether their attitude was positive or negative, and I believe logistic regression would be good to deal with classification problem. Then I put all those steps into a pipeline The following code is the ML pipeline:

tokenizer = Tokenizer(inputCol="clean\_review\_body", outputCol="clean\_review\_words")

remover = StopWordsRemover(inputCol="clean\_review\_words", outputCol="remove\_stopword")

hashtf = HashingTF(numFeatures=2\*\*16, inputCol="remove\_stopword", outputCol='tf')

idf = IDF(inputCol='tf', outputCol="features", minDocFreq=5)

label\_stringIdx = StringIndexer(inputCol = "rating\_converted", outputCol = "label")

lr = LogisticRegression(maxIter=100)

pipeline = Pipeline(stages=[tokenizer,remover, hashtf, idf, label\_stringIdx, lr])

Before I fit the pipeline to traning set, I tried to create grid search to find the most fitted hyperparameter which lead to best model performance.

grid = ParamGridBuilder()

grid = grid.addGrid(lr.regParam, [0.0, 1.0])

grid = grid.addGrid(lr.elasticNetParam, [0, 1])

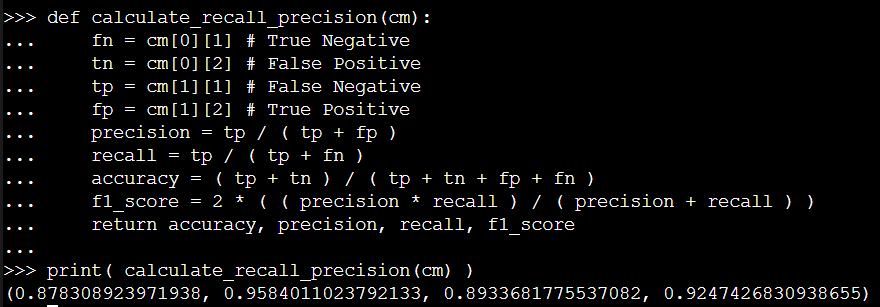
Here we would create 4 models to be tested. And we use crossvalidator with the hyperprameter grid with the evaluator AUC:

evaluator = BinaryClassificationEvaluator(metricName="areaUnderROC")

cv = CrossValidator(estimator=pipeline, estimatorParamMaps=grid,evaluator=evaluator,

numFolds=3, seed=789)

After those steps, I can fit the model to training set and transform to test set and to use some evaluation funcnction such as accuracy, precision, recall, f1 score.



Besides, we also have area under the ROC curve and confusion matrix, to see model performance, which will be showed in the visualization part. After evaluated the model, I save this file into parquet file then I convert it into csv to check some of the result and the prediction. The following are some of the results:



Above is the result of the prediction of shoes, it said “After only 4 months of wearing these shoes, the bottom sole split in half I would never but this again”. And is successfully classify as negative (1).



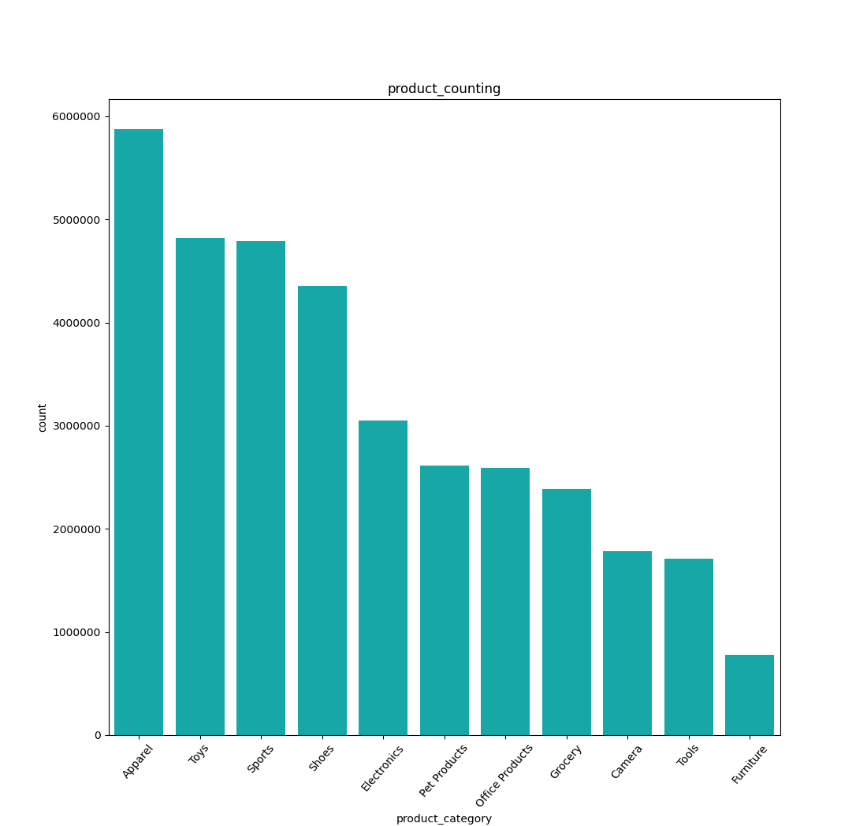
This one is kind of interesting since although they said “Love these”, but the rest of the sentence was talked about some flaws of the products, then the model classify it as negative (1).



Above are some other example of the positive (0) results.

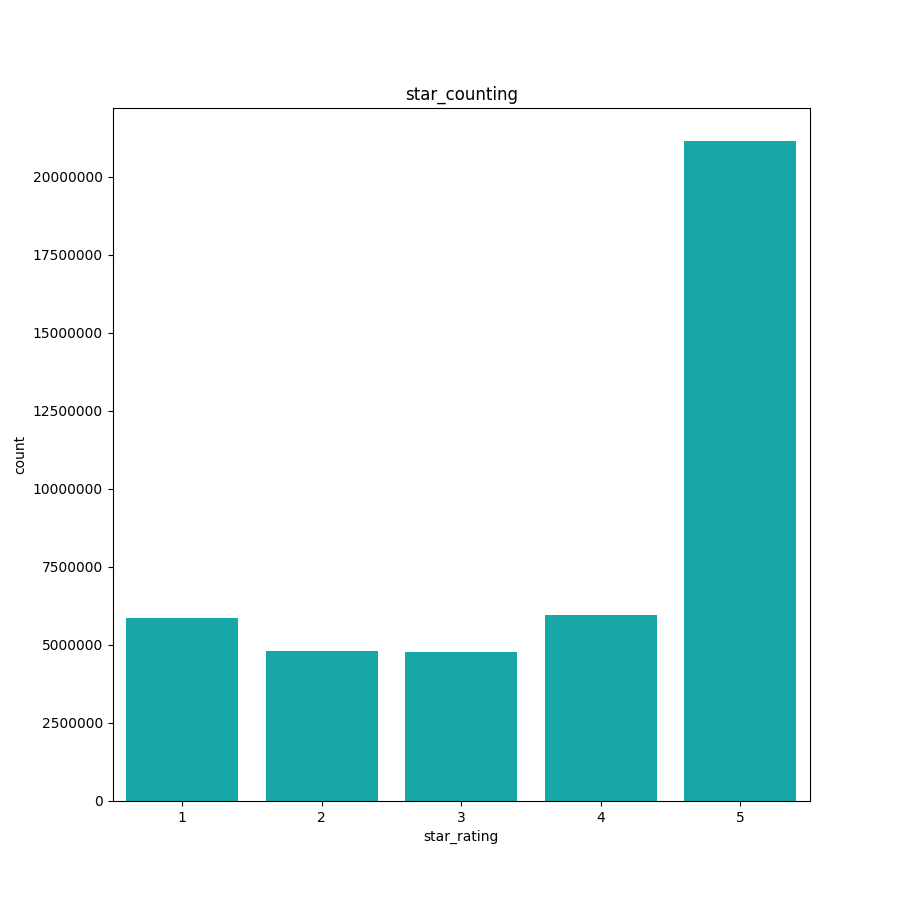
# Milestone 5: Visualization

1. Product counting graph (EDA)



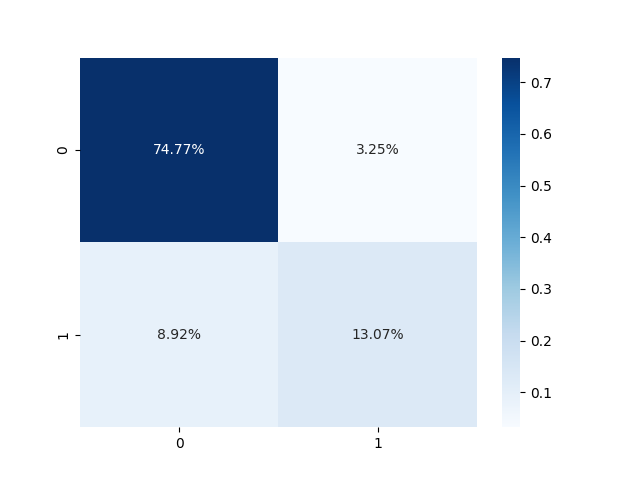
This is the bar chart created by function bar plot from seaborn. I find the all the distinct product category in the dataset, group by the product category and count each of the product in the dataset in descending. As we can see from the graph, the most product people shopped on Amazon is apparel and the least is furniture.

1. Star counting graph(EDA)



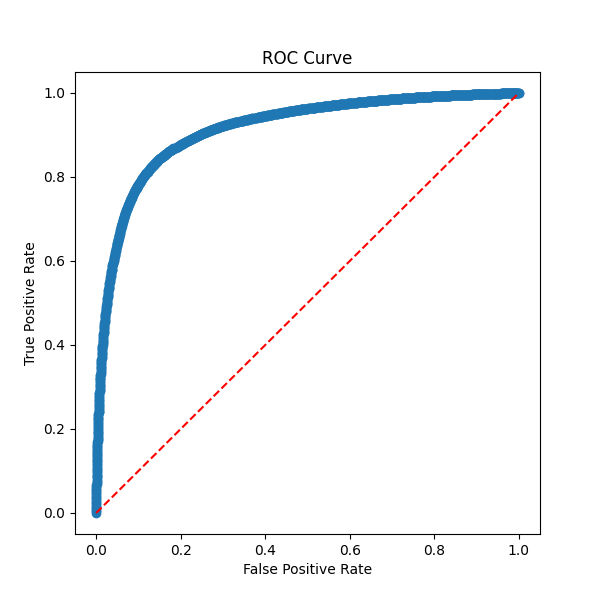
It is also the data exploratory graph which explored star rating range from 1 to 5. Obviously, we can find out that most of the reviews are 5 stars. Rating from 1 to 4 stars were relatively low in this dataset, then we can tell the quality of the products and service from Amazon are good and satisfied most of the customers.

1. Confusion matrix (Model performance)



The confusion matrix is to measure the classification model performance. The 0 here means positive and 1 is negative. As we can see that the true positive is 74.77 percent and true negative was 12.07 percent. They account for around 88 percent of the result, which means the model can correctly classify most of the customers attitude from their reviews.

1. ROC curve (Model performance)



The ROC curve is created by false positive rate and true positive rate. As we can see our roc curve apparently close to top-left corner, which means we have a good model here and we also have larger area under the curve(AUC). We calculate the AUC by following code:

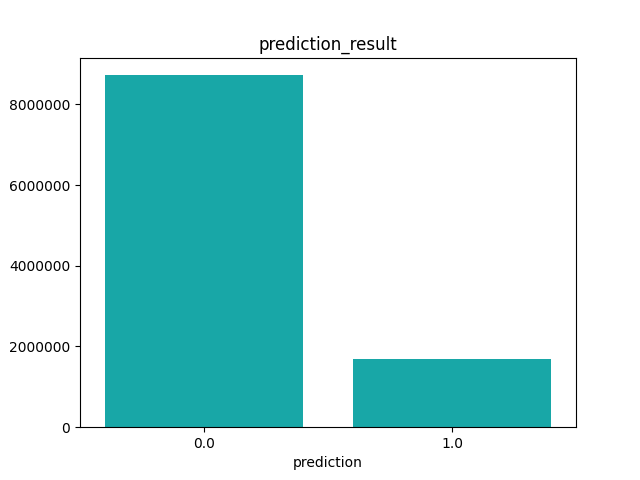
evaluator = BinaryClassificationEvaluator(metricName="areaUnderROC")

auc = evaluator.evaluate(predictions)

Text

Description automatically generated  
Since our AUC is between 0.90 to 1, I believe our model perform pretty well in this project.

1. Prediction result graph



This graph was the result of the prediction, which obviously can see most of the prediction was positive reviews. And this is what we can expect from the star counting graph since most of the reviews are 5 stars and it would refer that large fraction of result will be positive.

# Milestone 6: Summary and Conclusion

To sum up, we successfully classify the attitude from customers’ reviews, which started from finding the dataset, automating the process to download data from Kaggle to AWS S3, and do exploratory data analysis which extract some statistics of dataset. Then we start to clean the dataset, create our machine learning pipeline and used grid search and cross validator to create multiple models to help us to find most fitted hyperparameters to fit the trainset to produce the prediction results. In the end, we have some model evaluation such as confusion matrix, accuracy, precision, f1 score, recall, AUC and ROC curve then we create visualization both from EDA and model performance part and get some insight through the graphs. In my opinion, dealing with text data always need text-preprocessing process to remove those not important words to increase the accuracy of the model. Besides, we should better use hyperparameter tuning like grid search and cross validator to get a more accurate prediction result.

In conclusion, most of the result with obvious attitude such as “worst”, “nice”, “good”, “happy”,” unfortunately”, “sad “are easy to recognize by the model. And I found out that for those false positive and false negative results were because some of the reviews they complimented the products, but they also mentioned some drawbacks of products. It might be the future works for me to used other tools to work on specific cases.

# Appendix A: Code for downloading data

Region name – $us-east-2

Output format- $json

aws s3api create-bucket –mybucket4130 --region us-east-2 --create-bucket-configuration \ LocationConstraint=us-east-2

pip3 install kaggle

mkdir .kaggle

nano .kaggle/kaggle.json

chmod 600 .kaggle/kaggle.json

kaggle datasets list

nano ~/.local/lib/python3.7/sitepackages/kaggle/api/kaggle\_api\_extended.py

#change to these two lines

if not os.path.exists(outpath) and outpath != "-":

with open(outfile, 'wb') if outpath != "-" else os.fdopen(sys.stdout.fileno(), 'wb', closefd=False) as out:

kaggle datasets list

kaggle datasets download

kaggle datasets download --quiet -d cynthiarempel/amazon-us-customer-reviews-dataset-p - | aws s3 cp - s3://mybucket4130/amazonreview.zip

#Unzip the file

import zipfile

import boto3

from io import BytesIO

bucket="mybucket4130"

zipfile\_to\_unzip="amazonreview.zip"

s3\_client = boto3.client('s3', use\_ssl=False)

s3\_resource = boto3.resource('s3')

zip\_obj = s3\_resource.Object(bucket\_name=bucket, key=zipfile\_to\_unzip)

buffer = BytesIO(zip\_obj.get()["Body"].read())

z = zipfile.ZipFile(buffer)

# Loop through all of the files contained in the Zip archive

for filename in z.namelist():

print('Working on ' + filename)

# Unzip the file and write it back to S3 in the same bucket

s3\_resource.meta.client.upload\_fileobj(z.open(filename),

Bucket=bucket,Key= f'{filename}')

# Appendix B: Carrying out descriptive statistics

#data cleaning and statistics of data

from pyspark.sql.functions import col, isnan, when, count, udf, to\_date, year, month, date\_format, size, split

#read the data from s3 bucket

df=spark.read.csv('s3n://mybucket4130/final\_amz.csv',header=True)

#print all columns

df.columns

#print the summary of the dataframe

df.summary().show()

#check null values of star\_rating and review\_body column

df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in

["star\_rating", "review\_body"]] ).show()

#count how many records in the dataframe

df.count()

#drop null values

df = df.na.drop(subset=["star\_rating", "review\_body"])

#define function to drop the emoji from customer reviews

def ascii\_only(mystring):

if mystring:

return mystring.encode('ascii', 'ignore').decode('ascii')

else:

return None

# assign function to udf

ascii\_udf = udf(ascii\_only)

#applied function to review body

df = df.withColumn("clean\_review\_body", ascii\_udf('review\_body'))

#save the cleaned data to csv

output\_file\_path="s3://mybucket4130/ cleaned\_data.csv'"

df.write.options(header='True', delimiter=',').csv(output\_file\_path)

# Appendix C: ML pipeline (cleaning, feature extraction, model building)

#import all library we need

Import io

import pandas as pd

import s3fs

import boto3

import matplotlib.pyplot as plt

import seaborn as sns

from pyspark.sql.functions import col, isnan, when, count, udf, to\_date, year, month, date\_format, size, split

from pyspark.ml.stat import Correlation

from pyspark.ml.feature import VectorAssembler

from pyspark.sql.types import IntegerType

from pyspark.ml.feature import HashingTF, IDF, Tokenizer, StringIndexer

from pyspark.ml import Pipeline

from pyspark.ml.classification import LogisticRegression

from pyspark.ml.tuning import CrossValidator, ParamGridBuilder

from pyspark.ml.evaluation import BinaryClassificationEvaluator

from sklearn.metrics import confusion\_matrix

from pyspark.ml.feature import StopWordsRemover

# read the clean data

df=spark.read.csv('s3n://mybucket4130/cleaned\_data.csv',sep='\t', header=True, inferSchema=True)

#drop the column we don't need

df=df.drop('\_c0', 'marketplace', 'customer\_id', 'review\_id', 'product\_id', 'product\_parent', 'product\_title', 'product\_category','helpful\_votes', 'total\_votes', 'vine', 'verified\_purchase', 'review\_headline', 'review\_body', 'review\_date')

# check the null values in dataset

df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in

["clean\_review\_body","star\_rating"]] ).show()

#remove any non numeric value in star rating

df = df.filter(~df.star\_rating.rlike('\D+'))

# drop the rows with null values

df = df.na.drop(subset=["clean\_review\_body","star\_rating"])

#convert the star\_rating column from string to integer

df = df.withColumn("star\_rating",df.star\_rating.cast(IntegerType()))

#create a column to identify if the star rating >=3=1, otherwise is 0

df = df.withColumn("rating\_converted", when(col("star\_rating") > 3, 1).otherwise(0))

#split the dataset

train\_set, test\_set = df.randomSplit([0.7, 0.3], seed = 2000)

#convert the sentence to token

tokenizer = Tokenizer(inputCol="clean\_review\_body", outputCol="clean\_review\_words")

#remove stopwords

remover = StopWordsRemover(inputCol="clean\_review\_words", outputCol="remove\_stopword")

#convert words to vectors

hashtf = HashingTF(numFeatures=2\*\*16, inputCol="remove\_stopword", outputCol='tf')

#create inverse document frequency

idf = IDF(inputCol='tf', outputCol="features", minDocFreq=5)

#create indexer for the rating\_converted column

label\_stringIdx = StringIndexer(inputCol = "rating\_converted", outputCol = "label")

#create model

lr = LogisticRegression(maxIter=100)

#create pipeline

pipeline = Pipeline(stages=[tokenizer,remover, hashtf, idf, label\_stringIdx, lr])

# Create a grid to hold hyperparameters

grid = ParamGridBuilder()

grid = grid.addGrid(lr.regParam, [0.0, 1.0])

grid = grid.addGrid(lr.elasticNetParam, [0, 1])

# Build the parameter grid

grid = grid.build()

# How many models to be tested

print('Number of models to be tested: ', len(grid))

# Create a BinaryClassificationEvaluator to evaluate how well the model works

evaluator = BinaryClassificationEvaluator(metricName="areaUnderROC")

# Create the CrossValidator using the hyperparameter grid

cv = CrossValidator(estimator=pipeline,

estimatorParamMaps=grid,

evaluator=evaluator,

numFolds=3,

seed=789

)

# Train the models

cv = cv.fit(train\_set)

# Test the predictions

predictions = cv.transform(test\_set)

# Calculate AUC

evaluator = BinaryClassificationEvaluator(metricName="areaUnderROC")

auc = evaluator.evaluate(predictions)

print('AUC:', auc)

# Create the confusion matrix

predictions.groupby('label').pivot('prediction').count().fillna(0).show()

cm = predictions.groupby('label').pivot('prediction').count().fillna(0).collect()

def calculate\_recall\_precision(cm):

fn = cm[0][1] # False Negative

tn = cm[0][2] # True Negative

tp = cm[1][1] # True Positive

fp = cm[1][2] # false 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) )

# Appendix D: Visualization

**Save each graph through following code:**

img\_data=io.BytesIO()

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

img\_data.seek(0)

s3=s3fs.S3FileSystem(anon=False)

with s3.open('s3://mybucket4130/file\_name.png','wb') as f:

f.write(img\_data.getbuffer())

**visualization 1: product category couunting**

gg=df.groupby('product\_category').count().sort('count',ascending=False).show()

pd=['Apparel','Toys','Sports','Shoes','Electronics','Pet Products','Office Products','Grocery','Camera','Tools','Furniture']

sb=gg.filter(gg.product\_category.isin(pd))

sb.groupby('product\_category').count().sort('count',ascending=False).show()

qty=sb.groupby('product\_category').count().sort('count',ascending=False).toPandas()

grh=sns.barplot(x=qty['product\_category'],y=qty['count'],color='c').set(title='product\_counting')

grh.set\_xticklabels(grh.get\_xticklabels(),rotation=50)

plt.figure(figsize = (11,11))

plt.ticklabel\_format(style='plain',axis='y')

-----------------------------------------------------------------------------------------------------

**Visualization 2: star rating counting**

str=df.groupby('star\_rating').count().sort('count',ascending=False).toPandas()

grh=sns.barplot(x=str['star\_rating'],y=str['count'],color='c').set(title='star\_counting')

grh.set\_xticklabels(grh2.get\_xticklabels(),rotation=0)

-----------------------------------------------------------------------------------------------------

**Visualization 3: confusion matrix**

y\_true = predictions.select("label")

y\_true = y\_true.toPandas()

y\_pred = predictions.select("prediction")

y\_pred = y\_pred.toPandas()

cnf\_matrix = confusion\_matrix(y\_true, y\_pred)

sns.heatmap(cf\_matrix/np.sum(cf\_matrix), annot=True, fmt='.2%', cmap='Blues')

-----------------------------------------------------------------------------------------------------

**Visualization 4:ROC curve**

# Look at the parameters for the best model that was evaluated from the grid

parammap = cv.bestModel.stages[5].extractParamMap()

for p, v in parammap.items():

print(p, v)

# Grab the model from Stage 5 of the pipeline

mymodel = cv.bestModel.stages[5]

plt.figure(figsize=(5,5))

plt.plot(mymodel.summary.roc.select('FPR').collect(),

mymodel.summary.roc.select('TPR').collect())

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title("ROC Curve")

--------------------------------------------------------------------------------------------------------

**Visualization 5: prediction result**

qty=predictions.groupby('prediction').count().toPandas()

sns.barplot(x=qty['prediction'],y=qty['count'],color='c')

plt.ticklabel\_format(style='plain',axis='y')

plt.title('prediction\_result')

with s3.open('s3://mybucket4130/predict\_result2.png','wb') as f:

f.write(img\_data.getbuffer())

Sources:

1. <https://github.com/Kaggle/kaggle-api/issues/315>
2. [Sentiment Analysis with PySpark. One of the tools I’m deeply interested… | by Ricky Kim | Towards Data Science](https://towardsdatascience.com/sentiment-analysis-with-pyspark-bc8e83f80c35)
3. [Sentiment-Analysis-and-Text-Classification-Using-PySpark/Food Review.ipynb at master · shikha720/Sentiment-Analysis-and-Text-Classification-Using-PySpark · GitHub](https://github.com/shikha720/Sentiment-Analysis-and-Text-Classification-Using-PySpark/blob/master/Code/Food%20Review.ipynb)
4. [Confusion Matrix Visualization. How to add a label and percentage to a… | by Dennis T | Medium](https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea)