CIS 4130

Salman Khan

salman.khan4@baruchmail.cuny.edu

Proposal Milestone 1:

Link: https://www.kaggle.com/datasets/cynthiarempel/amazon-us-customer-reviews-dataset

Description:

This dataset shows a sample of customer evaluations and opinions (was constructed to represent a sample of customer evaluations and opinions, variation in the perception of a product across geographical regions, and promotional intent or bias in reviews.) The information found in this data set is very useful for researchers in different fields of language processing and machine learning since hundreds of millions of consumers have provided reviews on products within the last two decades.

Prediction

I will be predicting whether a review is above 3 stars based on the number of words and length of the text with whether or not the review was sponsored or was for a verified purchase.

Attributes:

15 classes of products all serve as attributes:

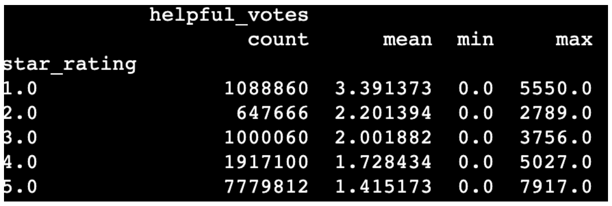
- marketplace
- customer_id
- review_id
- product_id
- product_parent
- product_title
- product_category
- star_rating
- helpful_votes
- total_votes
- vine
- verified_purchase
- review_headline
- review_body
- review_date

Milestone 2: Data Collection

For this milestone, the data was already stored for me in the AWS platform as a result, I did not need to download and collect the data into a bucket using a separate link. Instead I created an emr cluster and used a key pair with spark.

Milestone 3: Exploratory Data Analysis

```
Code for Milestone 3:

import boto3
import pandas as pd
s3 = boto3.resource('s3')
my_bucket = s3.Bucket('amazon-reviews-pds')


file_list = ['amazon_reviews_us_Home_v1_00.tsv.gz']


for fn in file_list:
    print("Working on file: ", fn)
    df = pd.read_csv('s3://amazon-reviews-pds/tsv/'+fn, 
sep='\t', compression='gzip', on_bad_lines='skip'  )
    df_all = df.append(df, ignore_index=True)
result= df_all.groupby('star_rating').agg({'helpful_votes':
['count','mean','min','max']})
print(result)
```



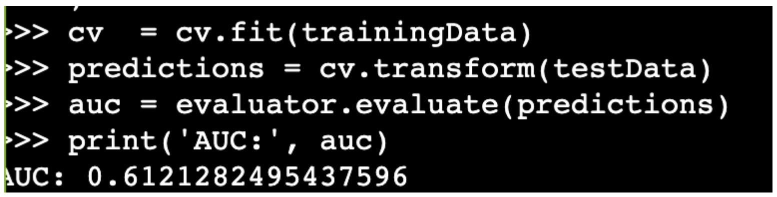
```
            helpful_votes
                    count      mean  min     max
star_rating
1.0               1088860  3.391373  0.0  5550.0
2.0                647666  2.201394  0.0  2789.0
3.0               1000060  2.001882  0.0  3756.0
4.0               1917100  1.728434  0.0  5027.0
5.0               7779812  1.415173  0.0  7917.0
```

Based on my data I see that many customers have given a 5 star rating based on the products they have purchased off of Amazon with over 7,779,812 reviews supporting it. The number of people who have given a 4 star review for their products is 1,917,100 reviews. These numbers are based on verified customers and purchases and have been labeled as “helpful comments” when checking the data.

Milestone 4: Coding and Modeling:

```
# Create a grid to hold hyperparameters
grid = ParamGridBuilder()
grid = grid.addGrid(lr.regParam, [0.0, 0.2, 0.4, 0.6, 0.8, 1.0])
grid = grid.addGrid(lr.elasticNetParam, [0, 0.5, 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=reviews_pipe,
                     estimatorParamMaps=grid,
                     evaluator=evaluator,
                     numFolds=3, seed=789 )
# Train the models
cv  = cv.fit(trainingData)
```

```
 positive_words = ["nice product", "will buy again", "great",
"amazing", "love it", "like it"]

sdf = sdf.withColumn("GoodWords", when(
lower(sdf.review_body).rlike('|'.join(positive_words)), 1.0
).otherwise(0.0) )
```



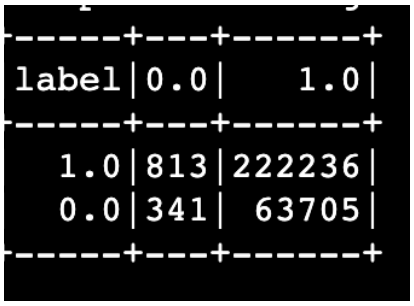
```
>>> cv  = cv.fit(trainingData)
>>> predictions = cv.transform(testData)
>>> auc = evaluator.evaluate(predictions)
>>> print('AUC:', auc)
AUC: 0.6121282495437596
```

Here I was able to create a grid and a parameter in which helps me to conduct my measuring given the dataset. The amount of models to be tested and how well the model I made works is also being tested here. Here I was able to conduct the accuracy rate of scoping for the customers that are verified consumers that actually purchased a product and left a review of higher than 3 stars. This came out to be 0.6121282495437596. Consumer feedback was also conducted based on simple compliments they left for a product as shown with the codes above.

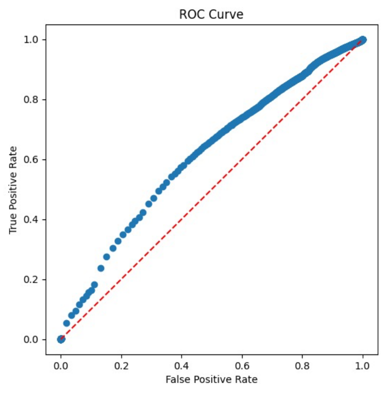
Milestone 5: Visualizing Results

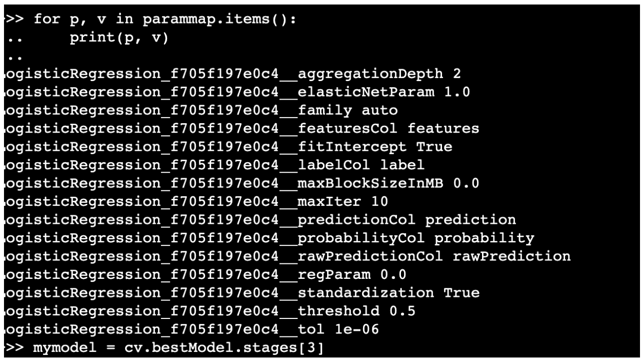
```
# Test the predictions
predictions = cv.transform(testData)

# Calculate AUC
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().fill
na(0).collect()

def calculate_precision_recall(cm):
tn = cm[0][1]
fp = cm[0][2]
fn = cm[1][1]
tp = cm[1][2]
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_precision_recall(cm) )

parammap = cv.bestModel.stages[3].extractParamMap()
for p, v in parammap.items():     print(p, v)
# Grab the model from Stage 3 of the pipeline
```

```
mymodel = cv.bestModel.stages[3]

import matplotlib.pyplot as plt
plt.figure(figsize=(6,6))
plt.plot([0, 1], [0, 1], 'r--')
x = mymodel.summary.roc.select('FPR').collect()
y = mymodel.summary.roc.select('TPR').collect()
plt.scatter(x, y)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title("ROC Curve")
plt.savefig("reviews_roc.png")
```

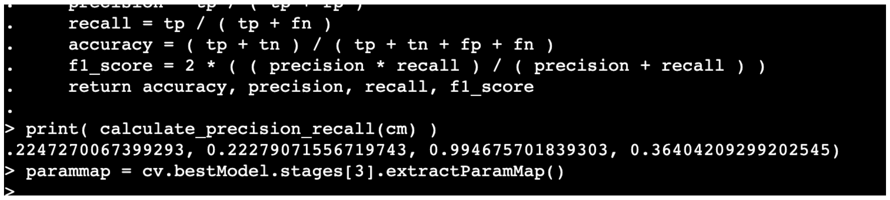


```
+-----+---+------+
 label|0.0|   1.0|
+-----+---+------+
   1.0|813|222236|
   0.0|341| 63705|
+-----+---+------+
```





```
>> for p, v in parammap.items():
..     print(p, v)
..
ogisticRegression_f705f197e0c4__aggregationDepth 2
ogisticRegression_f705f197e0c4__elasticNetParam 1.0
ogisticRegression_f705f197e0c4__family auto
ogisticRegression_f705f197e0c4__featuresCol features
ogisticRegression_f705f197e0c4__fitIntercept True
ogisticRegression_f705f197e0c4__labelCol label
ogisticRegression_f705f197e0c4__maxBlockSizeInMB 0.0
ogisticRegression_f705f197e0c4__maxIter 10
ogisticRegression_f705f197e0c4__predictionCol prediction
ogisticRegression_f705f197e0c4__probabilityCol probability
ogisticRegression_f705f197e0c4__rawPredictionCol rawPrediction
ogisticRegression_f705f197e0c4__regParam 0.0
ogisticRegression_f705f197e0c4__standardization True
ogisticRegression_f705f197e0c4__threshold 0.5
ogisticRegression_f705f197e0c4__tol 1e-06
>> mymodel = cv.bestModel.stages[3]
```



```
.     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_precision_recall(cm) )
.2247270067399293, 0.22279071556719743, 0.994675701839303, 0.36404209299202545)
> parammap = cv.bestModel.stages[3].extractParamMap()
>
```

Here I began testing my predictions and calculating my auc. Grouped by label and prediction as you see my result depicted in the image. This helps to depict the difference in finding what is a verified user purchase as compared to how accurately my data set was able to identify the users who actually left a review of higher than 4 stars. The ROC curve shows the true positive rate and the false positive rate and the results of my test that I ran is shown above.

Milestone 6: Summary & Conclusions

Based on this project that I did, I was able to run very interesting results that resulted in a positive outcome. The accuracy of the test I ran had resulted in a rate of 61.21282495437596. The amount of users who left an amazon rating of higher than 3 stars and are verified users who bought the product was accurately measured but not precisely 100%. Since my data set was already within the amazon web services site, I did not have to use a separate link and download it into a bucket within aws. Instead I ran an emr cluster which I then selected a pair key and used spark to run the data. When I imported my file, it allowed me to see the amount of helpful ratings I have received and are accurate as well as the amount for each rated star from 1-5. I then worked on creating a grid to hold hyperparameters and creating a parameter. I was able to conduct positive words that were left based on the most common words. In addition, the accuracy of my test that supports what I am trying to predict is supported here which resulted in 61.21282495437596. I then began working on visualizing results in order to better depict my findings using illustrations. This is where I began testing my predictions and calculating my auc. Here I am able to see the test results of the ROC curve which allows me to see the true positive rate and false positive rate of the results of my test. They were very closely aligned with the two lines not far away from each other at all. In conclusion, my prediction resulted in a positive outcome since it was able to pick up an accuracy rate that was well above 50%.

Milestone 7: Sharing the project

https://github.com/Salspizza00/cis_4130_project/blob/main/README.md