This assignment aims to show the use of crowd sourced data as an alternative of the gold standard data which is expensive to create and often unavailable. As this is a group assignment individual contribution has been mentioned at the end of this report.

1. Data Description

The following sections cover Task 1 and Task 4 and contain observations regarding the datasets regarding both task sections as well as explanations of the random sampling approach used in Part 1 – Task 1 and the Majority Vote implementation in Part 2.

1.1 Gold Sample (Task 1)

Dataset	Label Count	Comments
gold_sample.csv	1202	First column is id of the movie. Last column is gold standard expert classification review. Latent Semantic Analysis Features are specified in columns in between: 1200 features. Regarding feature labels, we have not applied any feature engineering i.e. reducing feature labels from total of 1200 as we do not know enough about the features as it just has the header TOPIC0-TOPIC1199, not some explicit identifier that we know we can remove from the training set (like a metadata ID).
	Feature Min Value	-2.24180373043. Will vary per script run unless random_state=1 (seed for random sample of 1000 records).
	Feature Max Value	1.89848313856. Will vary per script run unless random_state=1 (seed for random sample of 1000 records).

	Missing Values	We can see depending on results of the random sample for 1000 records from original gold.csv that sometimes we do have records in the data frame that contain missing values. A (not a number) NaN check is performed in the code and replaces NaN with zero if they occur in the sample, but there was little difference, if any, in model accuracy on the test set if such cases did occur.

1.2 Random Sampling

For the random sample of 1000 records task 1 which is also used as the input for task 4, we use pandas dataframe random sampling for the scripts to generate the required training sample file. This uses method with an input range N configured to be the size of the sample and the pandas dataframe sampling built-in function. The process is documented in the pandas documentation at this URL: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html

In gold.csv and gold_sample.csv, the labels were inspected to check total label counts by binary classification value in source gold.csv file and in the gold_sample.csv to ensure there was no heavy bias in the training sample towards one or other classification label, either positive or negative.

Below are the counts taken based on current settings and random_state = 1 on gold_sample.csv:

File	Label Type (Pos/Neg)	Count
gold.csv	Positive Labels	2502
gold.csv	Negative Labels	2497
gold_sample.csv	Positive Labels	509
gold_sample.csv	Negative Labels	491

1.3 MTurk Sample (Task 4)

Dataset	Label Count	Comments
mturk_sample.csv	1203	First column is id of the movie. Second column is annotator which we use as a synonym for “worker” throughout our Python code scripts; this column is crucial

		for subsequent tasks including majority vote and dawid-skene method and applying filters on the input files so we can filter by worker or generate a confusion matrix later for a worker. Last column is gold standard expert classification review. Latent Semantic Analysis Features are specified in columns in between: 1200 features. With regard to the features, we have not applied any feature engineering i.e. reducing feature labels from 1200 as we do not know enough about the features as it just has the header TOPIC0 to TOPIC1199.
	Feature Min Value	-4.62672002327. Will vary per script run unless random_state=1 (seed for random sample of 1000 records).
	Feature Max Value	5.033459236280001. Will vary per script run unless random_state=1 (seed for random sample of 1000 records).
	Workers	For task 4, we have “annotators” or workers. We have 190 unique workers in total. For each unique movie record/sentence id, we have on average, **5.559** workers (or annotators) per movie sentence id.

1.4 Majority Vote Implementation

The majority vote implementation uses a simple hard voting method of implementation; it does not use probabilities.

We pass in the movie identifier and the associated worker records for that movie id. We take a count of the positive ratings from all workers and a count of the negative rating for all workers. If positive counts outvote the negative counts, we set the sample record to be used to train model as a random sample from 1 or N of the worker samples used in the winning majority vote i.e. if 5 positive and 2 negative, we randomly

select from one of the 5 positive samples; that then becomes the training sample record later for the decision tree model in Part 2.

Similarly, for negative, we test if negative ratings outvote positive, and take a random sample from the 1 or N negative worker samples to be used later to train the decision tree.
Tie-breaker scenarios were tested as well. This simply checks if we have equality on the positive and negative worker review samples and if it is a 50:50 scenario, we just take a random sample from either the positive or negative worker samples as the sample record to use for training the decision tree.

Notes:

1) For the overall workflow of steps in each part, counts were verified for part 1, part 2, and part 3 using pages 57, 58, and 59 of the original presentation deck for the case study. So counts are cross-referenced against the corresponding step in the workflow diagram to ensure we are in same range in our files after the join and again when we save samples to file.
2) For the "pos" and "neg" string classification, as this is a binary classifier, we switched these to 1 and 0 for consistency to classify when a movie rating is positive or negative.

1.5 Dawid-Skene Implementation

The goal in this section was to build input training samples that contained best quality measure from the worker samples aggregated from the gold_sample.csv and the mturk.csv with the aggregated file being the input to the dawid-skene method.

The initial design iteration focussed on understanding the workflow on page 31 of the case studies presentation deck and initializing base polarity of the sample using a majority-vote baseline list and creating a confusion matrix per worker using out of the box scikit learn confusion matrix data structure and collating those matrices as a list per worker and movie sentence id to conceptualize the workflow.

The second design iteration took a from-scratch solution design approach and follows the steps 1-4 of the algorithm definition on page 31 of the slides. The code implementation then follows pages 32-43 so that the matrix data structures should mirror the data structures illustrated in the slides and the algorithm re-estimates polarity and probabilities until it converges and exits.

This output becomes the input for the decision tree training model discussed in section 2.

2. Machine Learning Model Analysis

The decision tree models are using scikit learn decision tree in Python.

2.1 Gold Sample and Decision Tree Model

The gold standard is the model we would have expected to perform the best as this is based on the domain expertise of trusted movie reviewers, so we expect the decision tree model should perform better based on this training sample.

Measurements: gold_sample.csv Model

Measure	**Result**
Accuracy	0.5434782608695652
F-Score	0.5299264214868271
Prediction Probabilities	Output to text file and used as input to plot the ROC curve.
True Positive Count (TP)	2112
False Positive Count (FP)	1872
True Negative Count (TN)	838
False Negative Count (FN)	606

Confusion Matrix based on TP/FP/TN/FN in preceding table:

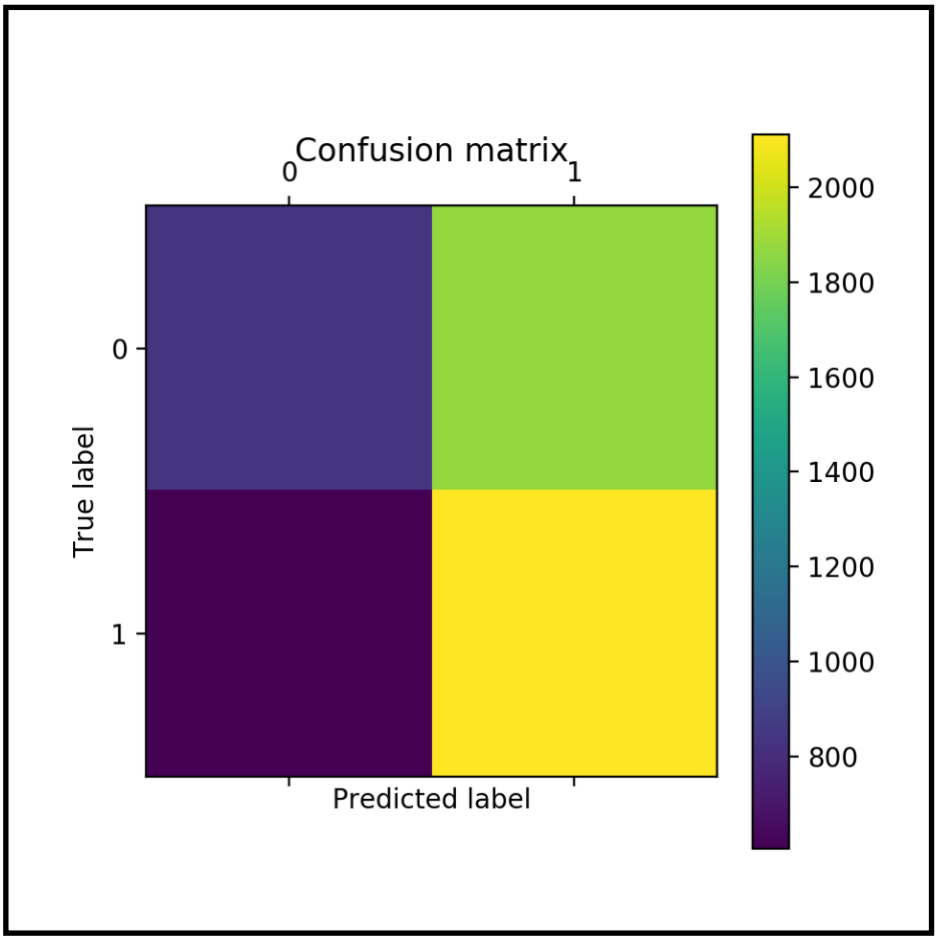

Figure 1: Confusion Matrix for gold_sample.csv decision tree model on test.csv

We can see the model has an almost equivalent number of *false positives (in particular) and false negatives*, undermining its prediction accuracy, indicating why we see only average overall model accuracy of 54-56%.

The following table illustrates the data for the F-Score which relies on precision and recall metrics. (The support column represents the number of samples of the true response that are in that category from test.csv):

Label	**Precision**	**Recall**	**F-Score**	**Support**
0	0.58	0.31	0.40	2710
1	0.53	0.78	0.63	2718
Micro Avg	0.54	0.54	0.54	5428
Macro Avg	0.56	0.54	0.52	5428
Weighted Avg	0.56	0.54	0.52	5428

2.1.1 RoC Curve Gold Sample Part 1 Model

The following shows the RoC curve for part 1 based on performance against test.csv. It plots the True Positive rate (y-axis) against the False Positive rate(x-axis). The False Positive rate is high so the orange curve line is being pulled away from the top-left corner which is our goal state for the Part 1 binary classifier. The area under roc (AUROC) is about 54%.

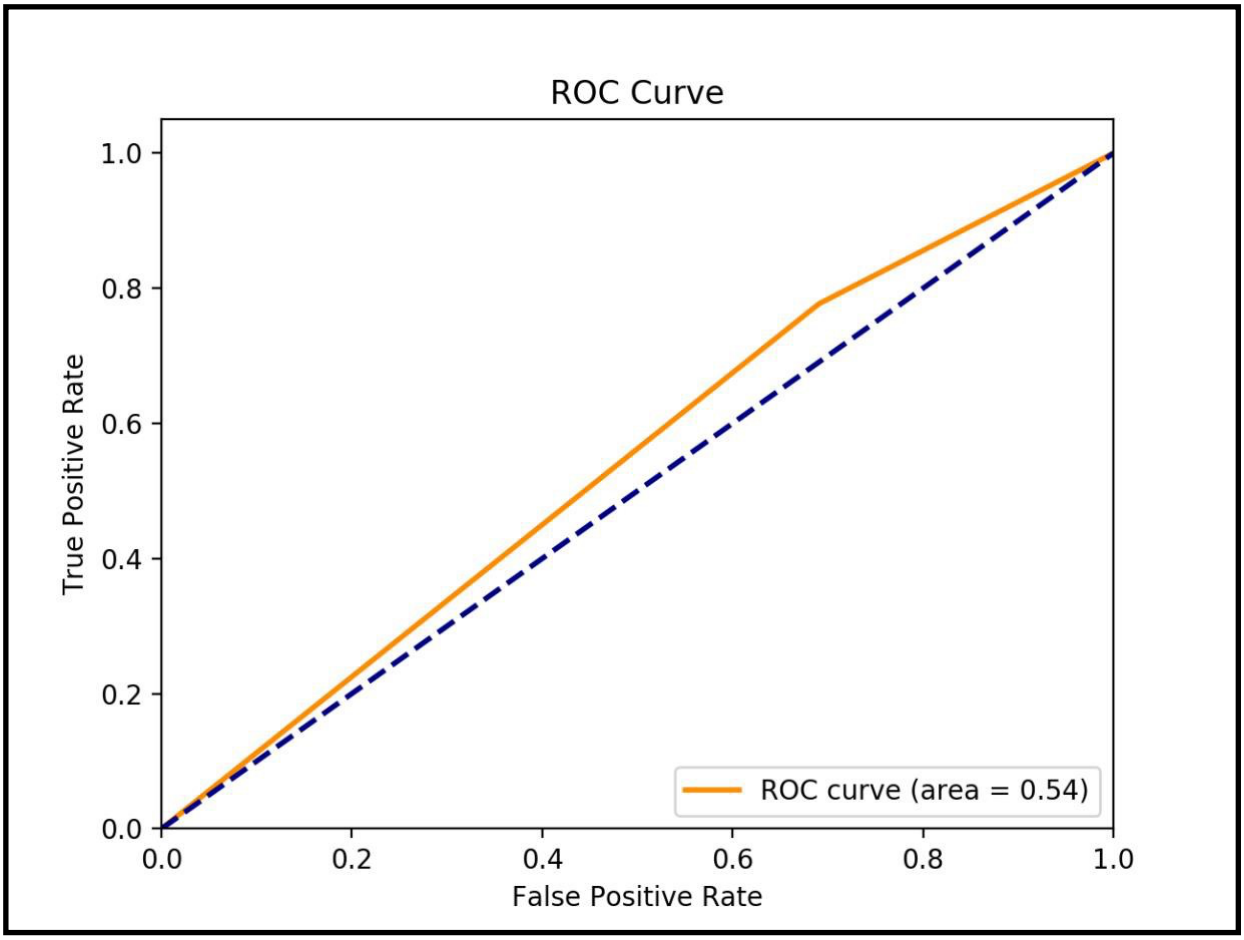

Figure 2: Part 1 Model RoC Curve Plot for True Positive against False Positive predictions

2.1.2 Decision Tree Graph for Model 1 Gold Sample

The following diagram illustrates the actual decision tree being plotted by the scitkit learn decision tree classifier, which is important to explain why the Decision Tree classifies as it does – the X[] values correspond to the TOPIC features for the latent semantic analysis values in the test.csv file. Three arguments (*max_depth=3, min_samples_split=2, min_samples_leaf=1*) only are configured to general guideline values used in the decision tree argument values in our Python code based on advice in the scikit learn documentation on using decision tree at this URL: https://scikit-learn.org/stable/modules/tree.html#tips-on-practical-use

The same settings are maintained for consistency in Part 2 and Part 3 decision tree configuration, so test cases are equivalent and to avoid one model using tuning parameters that are absent or different in another one.

We took the starting values suggested in the scikit learn documentation and adapted slightly until performance maxed out at 54%-56% or degraded back to 50-52% accuracy range.

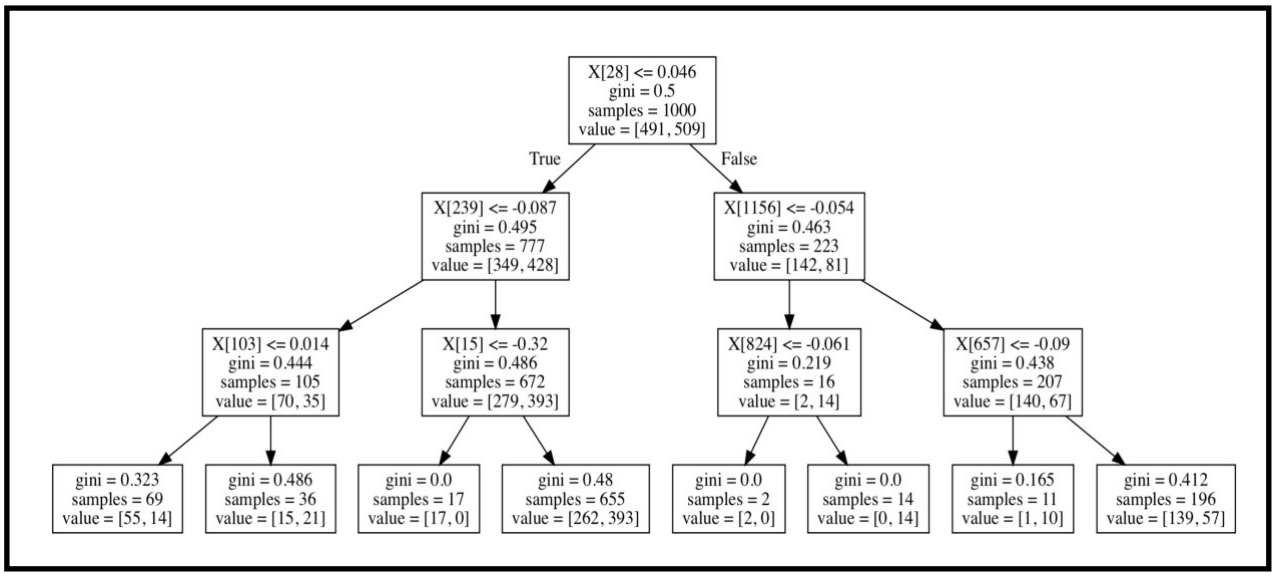

Figure 3: Decision Tree Graph for Part 1 Gold Sample Model

2.2 Majority Voting and Decision Tree Model

The majority vote model does not perform as well as the gold_sample model, but the reduction in performance is marginal at minus 2% to minus 4% less than the Part 1 Model per test run.

In our case study, it is known that sometimes crowdsourcing can produce near equivalent results to expert gold samples, so the result is not unexpected, but based on how our majority vote implementation is done, we only use a hard vote technique combined with a random sample from majority vote worker samples, so we know we may not always be getting the *best* worker sample record as the input record for the training set for Model 2 and a gold expert sample would likely be a better classification training sample for the decision tree model.

Measurements: mturk_sample.csv Majority Vote Model

Measure	**Result**
Accuracy	0.5147383935151069
F-Score	0.4798379791148015

Prediction Probabilities	Output to text file and used as input to plot the ROC curve.
True Positive Count (TP)	2227
False Positive Count (FP)	2143
True Negative Count (TN)	567
False Negative Count (FN)	491

Confusion Matrix based on TP/FP/TN/FN in preceding table:

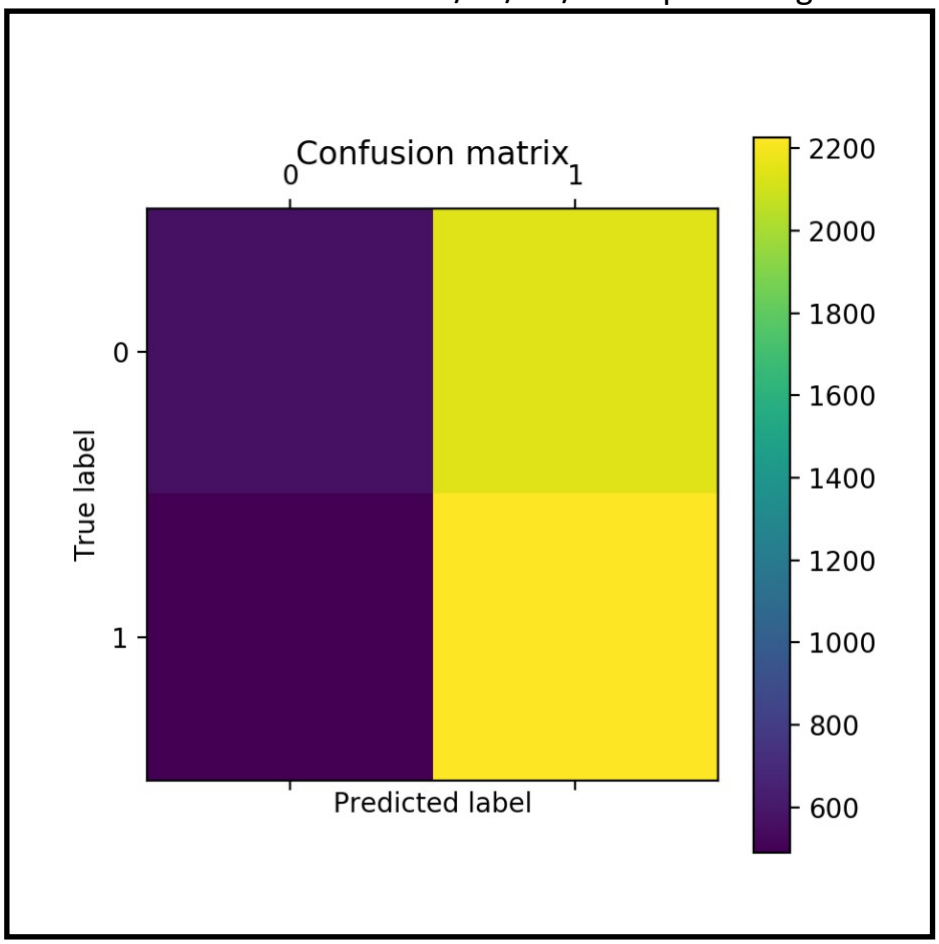

Figure 4: Confusion Matrix for mturk_sample.csv decision tree model on test.csv

We can see the model again (like in Model for Part 1) has a similar high number of *false positives and false negatives*, undermining its prediction accuracy, indicating why we see only average overall model accuracy of 51.4%.

The following table illustrates the data for the F-Score which relies on precision and recall metrics. (The support column represents the number of samples of the true response that are in that category from test.csv):

Label	**Precision**	**Recall**	**F-Score**	**Support**
0	0.54	0.21	0.30	2710

1	0.51	0.82	0.63	2718
Micro Avg	0.51	0.51	0.51	5428
Macro Avg	0.52	0.51	0.46	5428
Weighted Avg	0.52	0.51	0.46	5428

2.2.1 RoC Curve Gold Sample Part 2 Model

The following shows the RoC curve for part 2 based on performance against test.csv. It plots the True Positive rate (y-axis) against the False Positive rate(x-axis). The False Positive rate is high again, so the orange curve line is being pulled away from the top-left corner which is our goal state for the Part 1 binary classifier. The area under roc (AUROC) is about 51.42%.

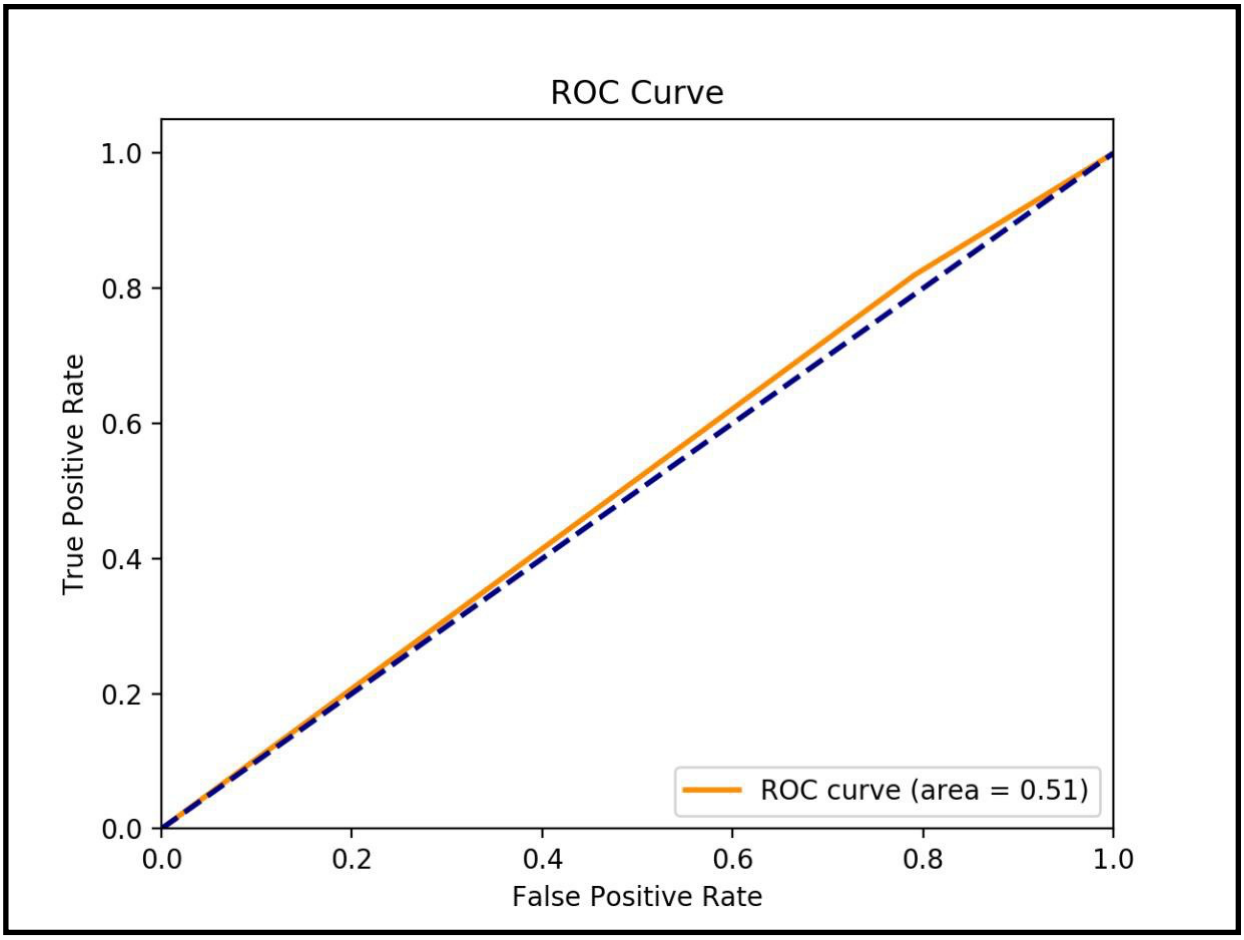

Figure 5: Part 2 Model RoC Curve Plot for True Positive against False Positive predictions

2.2.2 Decision Tree Graph for Model 2: Majority Vote

The following diagram illustrates the actual decision tree being plotted by the scitkit learn decision tree classifier, which is important to explain why the Decision Tree classifies as it does – the X[] values correspond to the TOPIC features for the latent semantic analysis values in the test.csv file.

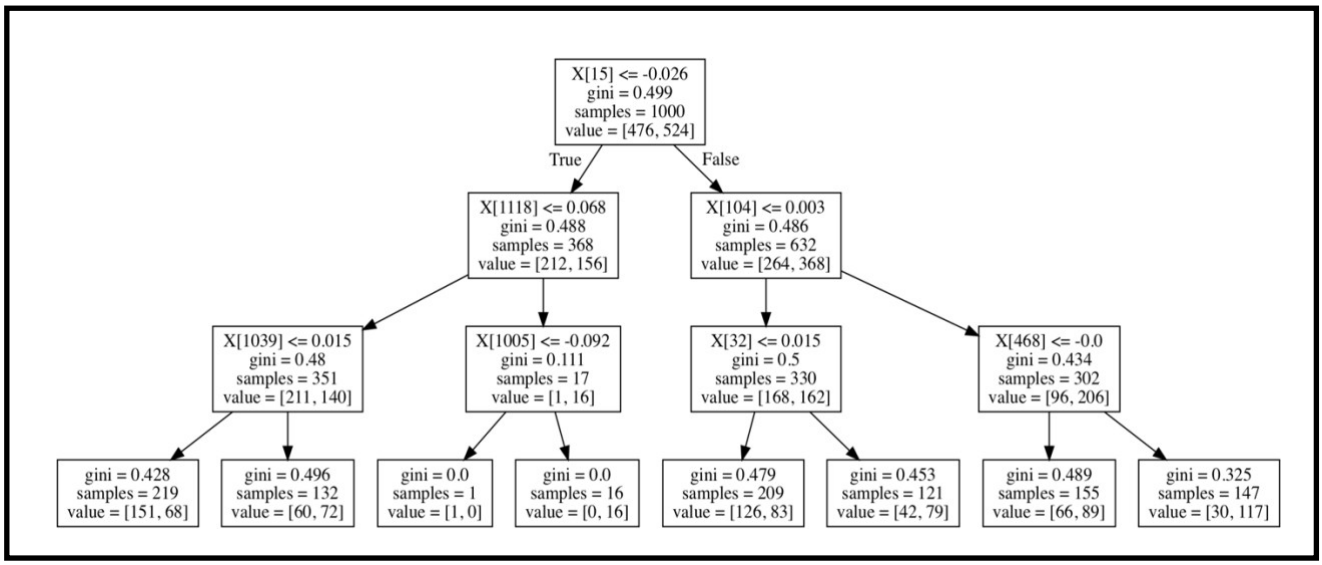

Figure 6: Decision Tree Graph for Part 2 Majority-Vote-driven Model

2.3 Dawid-Skene Method and Decision Tree Model

The dawid-skene method looks at more accurately predicting the probability of finding the best worker – the goal is to get the best quality worker and use them as the training sample input versus the simpler majority vote function from the model in Part 2. If this is performing correctly, we do expect better quality training samples than those in Part 2 Model, so it should perform better than the model in Part 2. This is the result we find during repeat unit testing on the dawid-skene method decision tree classifier results.

Measurements: dawid-skene Model

Measure	**Result**
Accuracy	0.5475313190862195
F-Score	0.5469580323054455
Prediction Probabilities	Output to text file.
True Positive Count (TP)	1627
False Positive Count (FP)	1365
True Negative Count (TN)	1345
False Negative Count (FN)	1091

Confusion Matrix based on TP/FP/TN/FN in preceding table:

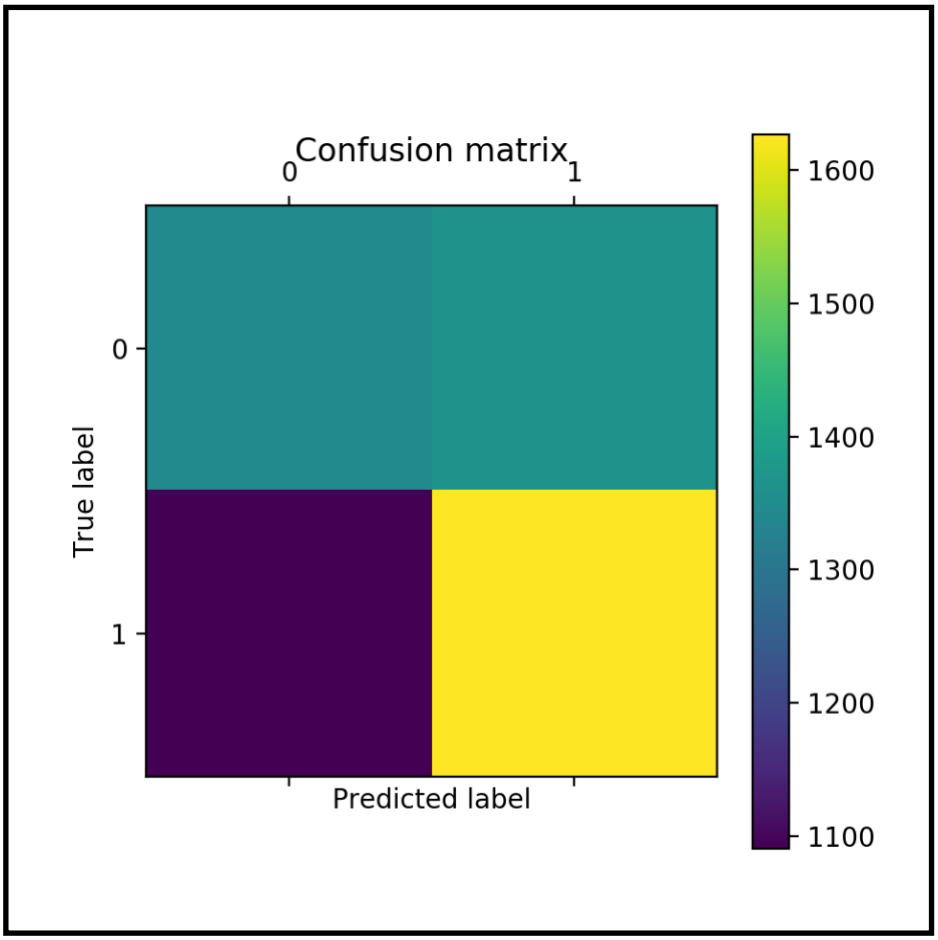

Figure 7: Confusion Matrix for dawid-skene decision tree model on test.csv

The following table illustrates the data for the F-Score which relies on precision and recall metrics. (The support column represents the number of samples of the true response that are in that category from test.csv):

Label	**Precision**	**Recall**	**F-Score**	**Support**
0	0.55	0.50	0.52	2710
1	0.54	0.60	0.57	2718
Micro Avg	0.55	0.55	0.55	5428
Macro Avg	0.55	0.55	0.55	5428
Weighted Avg	0.55	0.55	0.55	5428

2.3.1 RoC Curve for Part 3 Model

The following shows the RoC curve for part 3 based on performance against test.csv. It plots the True Positive rate (y-axis) against the False Positive rate(x-axis). The False Positive rate is high so the orange curve line is being pulled away from the top-left

corner which is our goal state for the Part 1 binary classifier. The area under roc (AUROC) is about 55%.

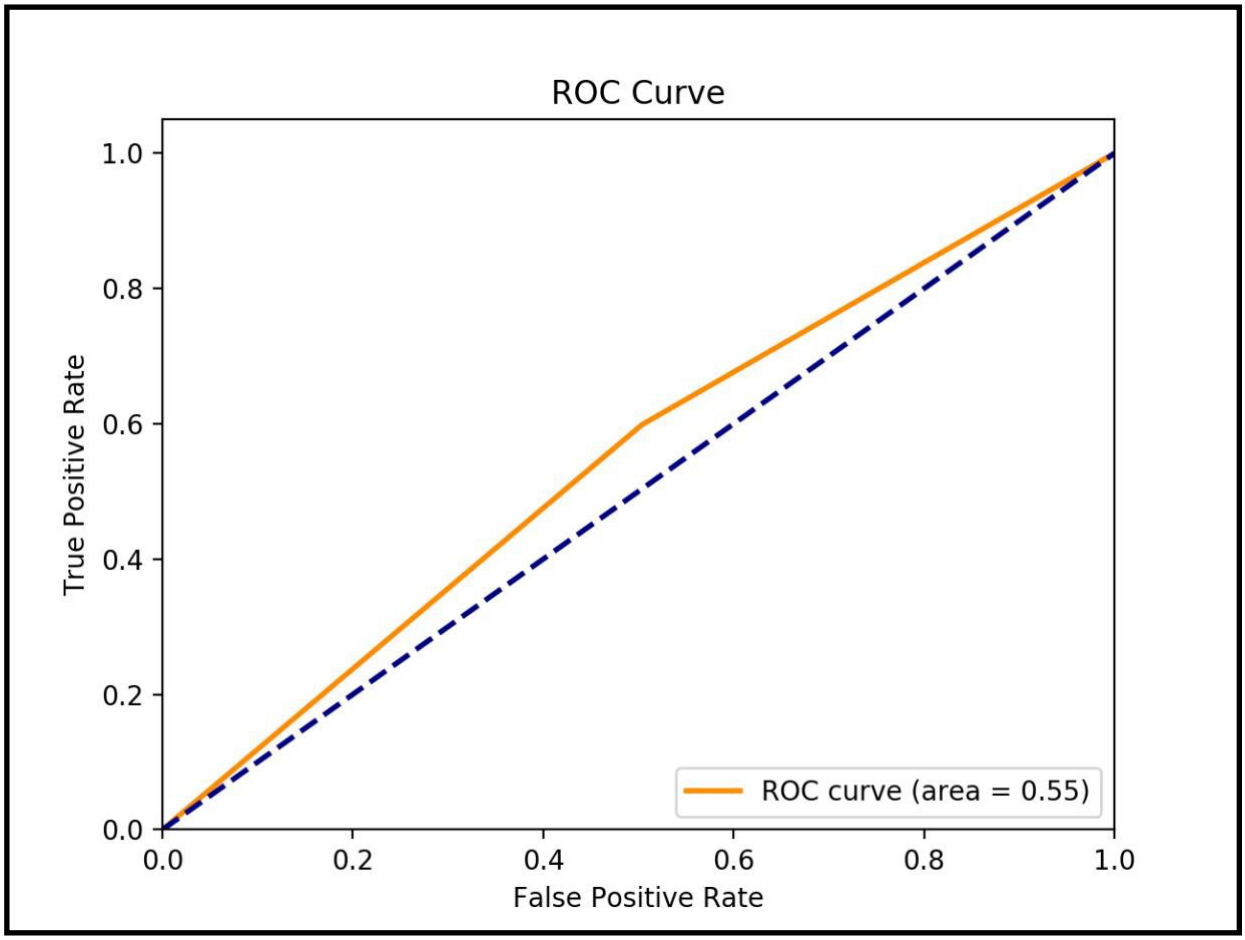

Figure 8: Part 3 Dawid-Skene Model RoC Curve Plot for True Positive against False Positive predictions

2.3.2 Decision Tree For Davin Skene

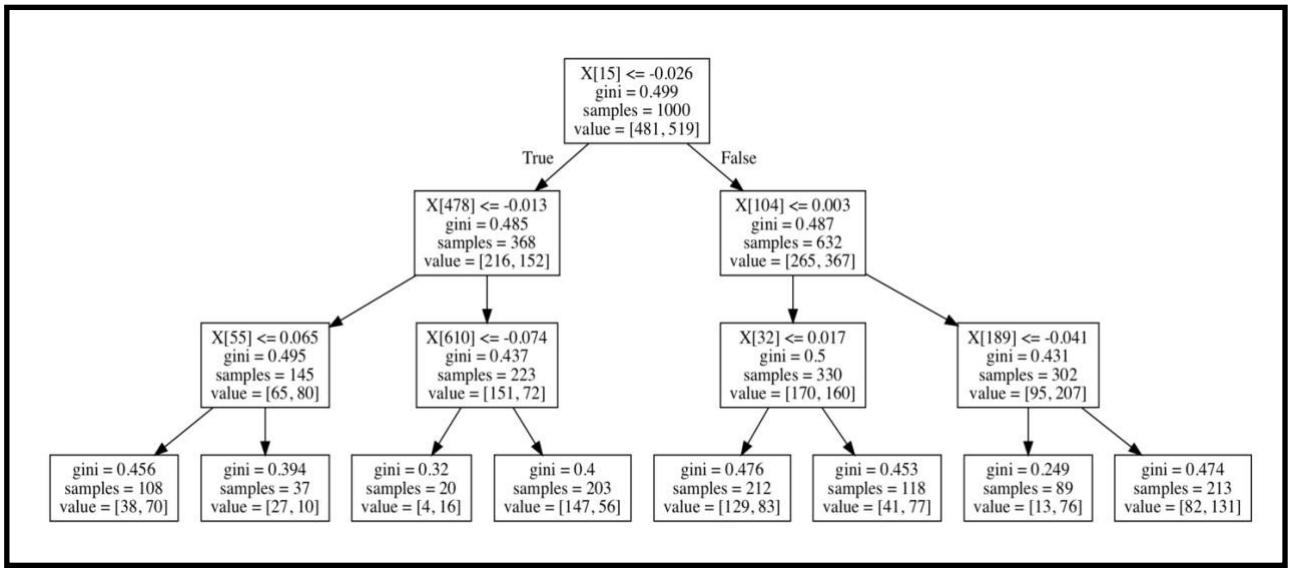

Figure 9: Decision Tree Graph for Part 3 dawid-skene method-driven Model

Observations :

- We have seen Gold Standard data gives the highest accuracy
- Majority Vote accuracy is less than the Gold Standard
- After implementing David Skene method, the accuracy increased from majority vote though its still low compared to gold standard data
- So, we can see as creating gold standard data is expensive, we can use crowd sourced data as an alternative

Python Library used in Coding -

- from sklearn import tree
- from sklearn.metrics import fbeta_score
- from sklearn.metrics import accuracy_score
- from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve, auc
- import pandas as pd
- import matplotlib.pyplot as plt
- import graphviz
- import numpy as np

Contribution:

- gold_sample.py - Niall Guerin
- train_gold.py - Niall Guerin
- mturk_sample.py - Niall Guerin
- train_mv.py - Niall Guerin & Shubhajit Basak
- train_ds.py - Shubhajit Basak
- Report - Niall Guerin & Shubhajit Basak

Bibliography

- Ul Hassan, U. (2019). *Classifying Movie Reviews Using Crowd-powered Machine Learning*.
- Dawid, A. and Skene, A. (1979). Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, [online] 28(1), pp.20-28. Available at: https://www.jstor.org/stable/2346806 [Accessed 25 Feb. 2019].
- Benalexkeen.com. (2019). *Scoring Classifier Models using scikit-learn – Ben Alex Keen*. [online] Available at: http://benalexkeen.com/scoring-classifier-models-using-scikit-learn/ [Accessed 1 Mar. 2019]. (Used same version and varied slightly and dropped R-type ggplot format he was using in his example)