Problem Statement

Classify Images into one of the 16 classes as shared in the data samples.

Requirements:

- Convert Images into Text and Location Using OCR
- Classify Images using Text and Location
- Use Deep Learning Models and Compare the findings.

Solution:

The dataset contained images in respective class folders, so it is safe to assume to go with the Supervised Learning task, also in our investigation we have seen that the Supervised Learning Classification approach outperforms the Unsupervised Clustering approach.

The proposed solution solves the Multi-Class Classification problem with XGBOOST. This also involves creating features from the images, texts extracted from the images. Pre and post-processing steps are also involved to get refined data for training.

Approaches:

1. Text Based Clustering

This involves experimentation with text based clustering utilizing base Hashing Vectorizer. Along with this, we used ngrams to create text embeddings for our use case. Unlike traditional vectorizers such as Countvectorizer or TF-IDF Vectorizer, Hashing Vectorizer does not maintain an explicit mapping between terms (words) and indices. Instead, it uses a hash function to directly convert terms into indices. This helps to create custom embeddings based on tokens/phrases which might not be present in vocab.

Approach:

Using Hashing Vectorizer for embeddings, Knee Elbow Method to understand optimal clusters, TSNE for dimensionality reduction, and KMeans for clustering.

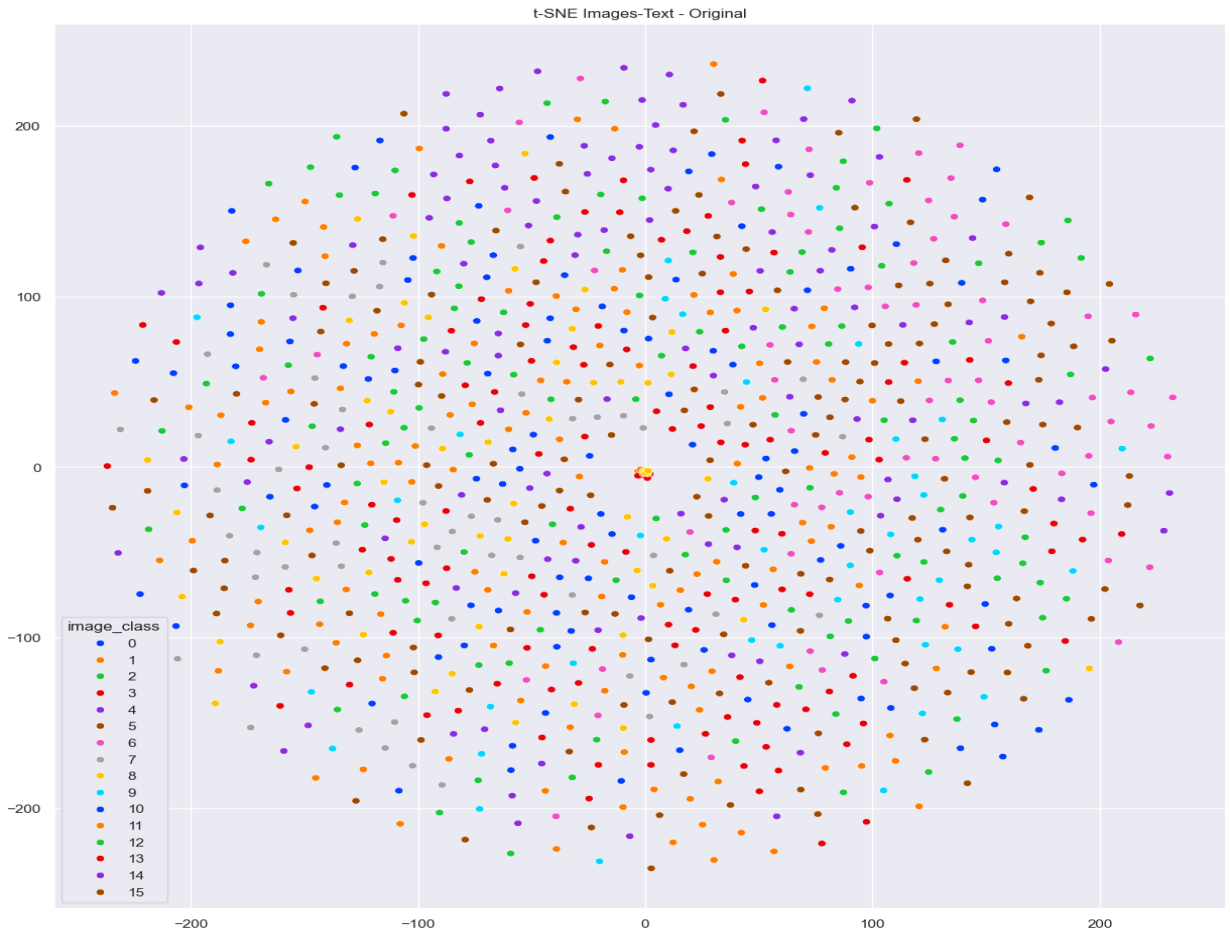
Pros:

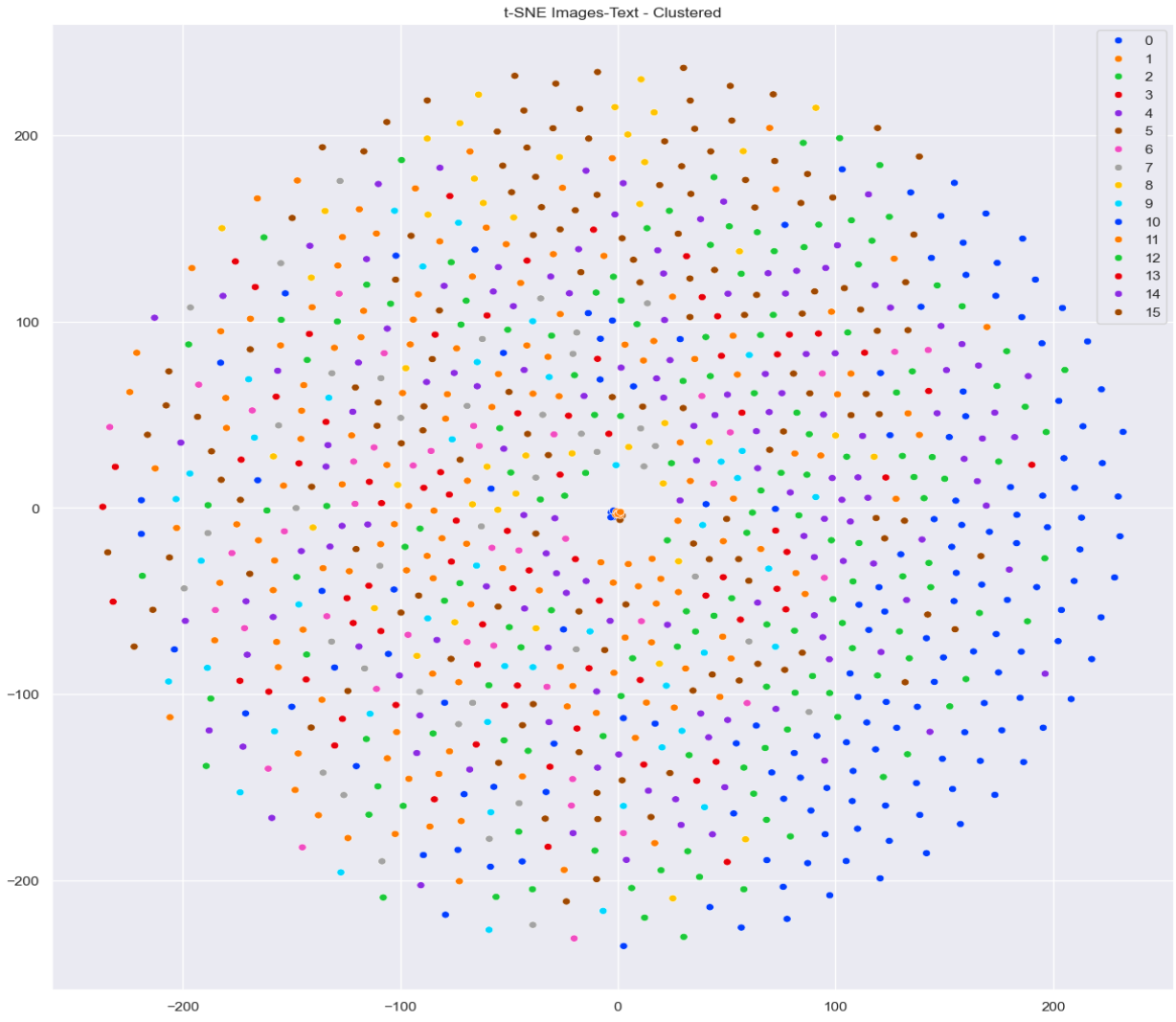
Helps with initial understanding of the text importance for given classes and predicted classes.

Cons:

This method is not helpful for this scenario, as the clusters formed based on the text appear dispersed and do not exhibit cohesive groupings.

Actual Labels VS Predicted Labels


t-SNE Images-Text - Original
200
100
0
−100
−200
−200
−100
0
100
200
image_class
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15



2. **Classification - with Transformer Embeddings , and Profiled Features.**

This approach incorporates principles from supervised learning. It entails generating features derived from text extracted from images and employing a pipeline to assess the performance across various classification models. Ultimately, XGBoost is employed as the final model for predicting classes.

Approach:

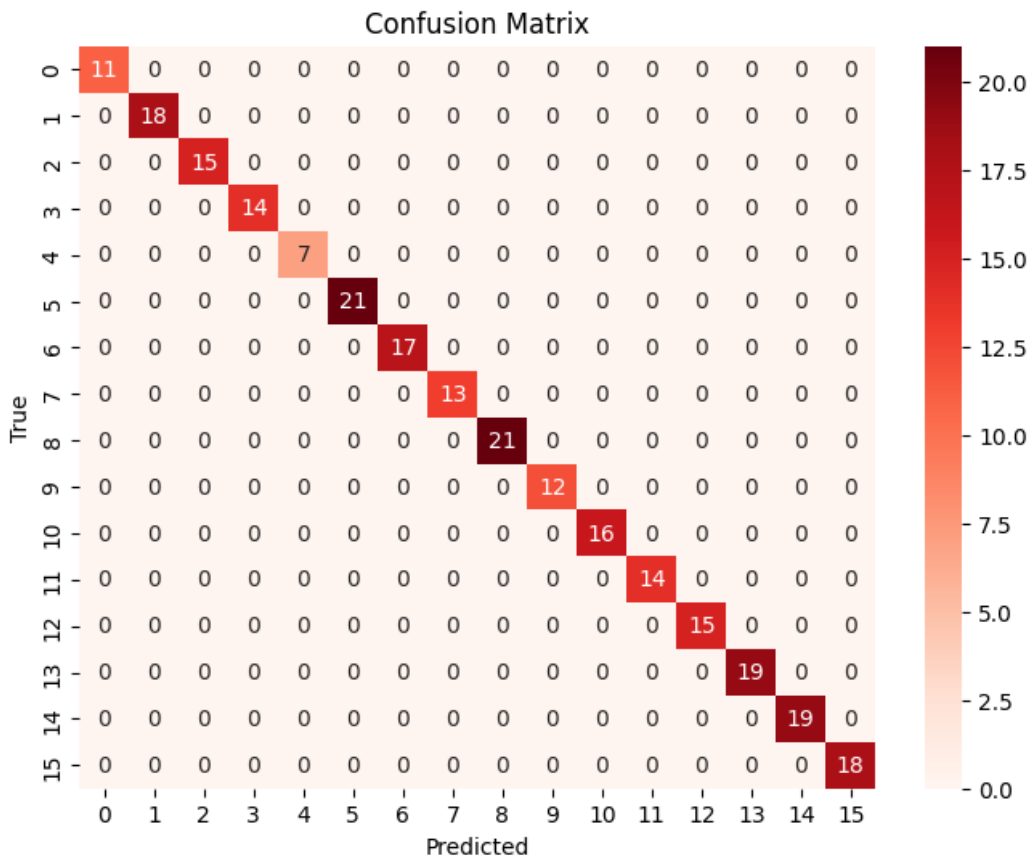
This approach uses Transformer based Embedding based on the model - ‘microsoft/MiniLM-L12-H384-uncased’ and XGBoost for multiclass classification. We are normalizing the embedding vector and creating 2D Features for the model to consume.

Evaluation Metrics:

R2 Score: 1.0
F1 Score: 1.0
Accuracy: 1.0
Precision: 1.0
Recall: 1.0

Pros:

1. Embeddings are contextually represented, it can result in effective feature generation as we have also seen in the Feature Importance of the Models(Less number of features which are not contributing to making decisions).
2. XGBoost is known for its robustness and effectiveness in handling complex classification tasks, we can scale the solution with faster training and experimentations.
3. This architecture is versatile in nature, we can finetune the Features as per our analysis, decide on how we want to draw our decision boundaries.



Cons:

1. Interpreting high dimensional data , i.e. embedding from the transformer model by XGBoost model can be challenging.
2. As we are using a pretrained model, which gives flexibility to work with broader vocab but our model's performance might highly depend on the model.
3. Training Data Quality significantly impacts the models performance.
4. High Number of features are not being utilized for models performance, but it is less than Approach 3, this can be optimized with further experimentation.

```
[32]  feats_importance[feats_importance['score']==0]
```

	feature_name	score
1	characters_count	0.0
4	duplicates_count	0.0
5	embed_feature_1	0.0
7	embed_feature_100	0.0
8	embed_feature_101	0.0
...	...	...
390	noun_phase_count	0.0
391	punctuations_count	0.0
392	sentences_count	0.0
394	sentiment_subjectivity_score	0.0
395	spelling_quality_score	0.0

212 rows × 2 columns

3. **Classification - with FastText Embeddings , and Profiled Features.**

This approach is almost similar to the previous approach, the difference here is we are using FastText Embeddings.

Approach:

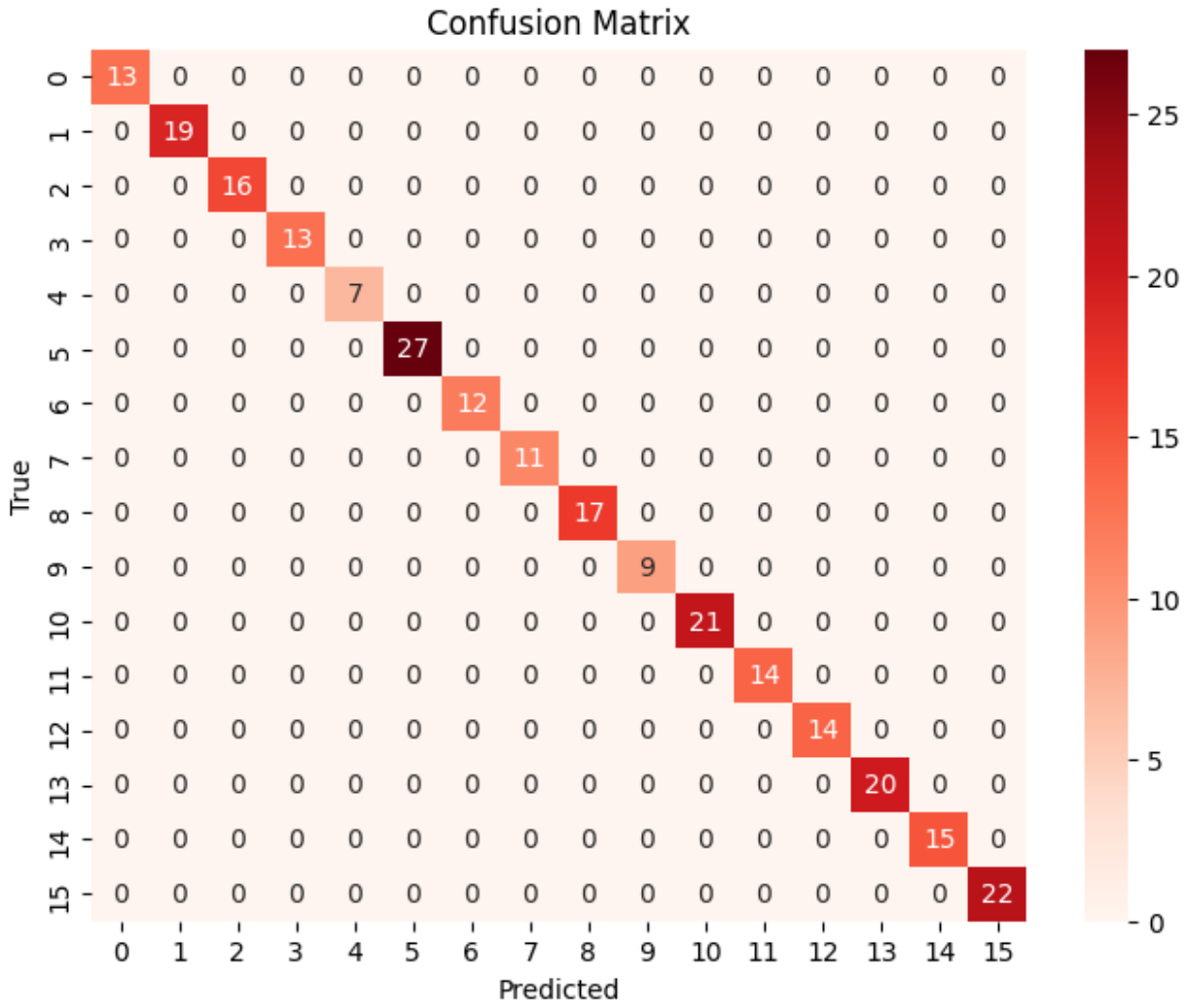
This approach uses Pretrained Model from Gensim - 'fasttext-wiki-news-subwords-300'
For creating embeddings and utilizing its features for training multiclass classification models using XGBoost.

Evaluation Metrics:

R2 Score: 1.0
F1 Score: 1.0
Accuracy: 1.0
Precision: 1.0
Recall: 1.0

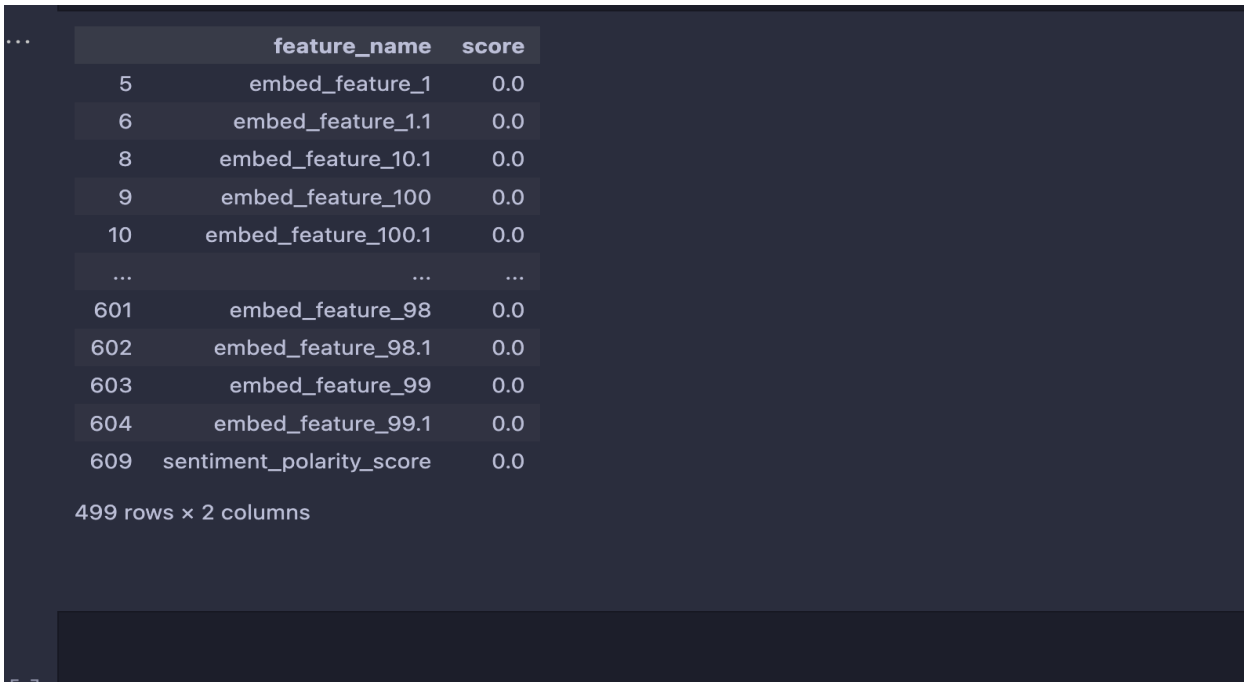
Pros:

1. The pretrained model might have unique rare words which could be useful.
2. FastText Embeddings can capture semantics and rare n-grams which can help for edge-cases.



Cons:

1. The Embeddings are not as contextual in nature as the Transformer models.
2. Interpretation of embedding can be Challenging for the XGBoost Model.
3. Models performance is highly dependent on the Embeddings.
4. Higher numbers of features are not being utilized for model performance.



	feature_name	score
5	embed_feature_1	0.0
6	embed_feature_1.1	0.0
8	embed_feature_10.1	0.0
9	embed_feature_100	0.0
10	embed_feature_100.1	0.0
...	...	...
601	embed_feature_98	0.0
602	embed_feature_98.1	0.0
603	embed_feature_99	0.0
604	embed_feature_99.1	0.0
609	sentiment_polarity_score	0.0

499 rows × 2 columns

Final Approach:

Approach 2 and 3 are very similar in nature and also giving similar results , but approach 2 has an edge over approach 3 as it uses contextual embeddings rather than keyword based embedding.
Approach 2 has a high number of features which is contributing to models performance, whereas most of the features of approach 3 are not getting utilized.

We might see more observations if the train dataset is increased .

Observed:

1. Both the models are getting most impacted by the features created by the nlp-profiler such as word count, sentiment score, alpha numeric count and more.
2. Definitely the models are getting overfitted as the accuracy of the models are 100 percent, which we imperfect in real world solutions.

Considerations:

1. We want to classify the images based on the classes assigned from the training dataset samples.
2. We have created a test-train split with a 75-25 ratio.

Assumptions:

1. Textual Features hold more importance than Image/Pixel associated Features.

Appendix :

Listing some useful links for future work:

- Clustering : Clustering sentence embeddings to identify intents in short text | by David Borrelli | Towards Data Science
- Thresholding : Comparative Study on Threshold Techniques for Image Analysis (ijert.org)
- Thresholding Algorithms : Niblack and Sauvola Thresholding — skimage 0.22.0 documentation (scikit-image.org)
- Adding Image Embeddings : Image embeddings. Image similarity and building… | by Romain Beaumont | Medium
- Concept Image Embeddings : Improve Model Performance with Image Embeddings | Encord
- Using LLMs for Image Classification: 2311.11904.pdf (arxiv.org)