DATS-6103: Data Mining Final Project Group Report

Popular Attraction/Landmark Recognition Using Google Landmark Dataset

Introduction:

With a rapid increase in the use of smartphones and other social apps, Image Recognition, Image Classification and Image Processing are the latest concepts that interest data engineers in computer vision tasks. A major challenge with image classification is the lack of a large, annotated dataset to train better and robust models. Recognizing and training the model to identify any landmark is a challenging task as the appearance of the landmark varies with geometry, illumination and a different aspect ratio of the image presented. To overcome this issue, a collection of images is used to capture typical appearance of the location. This project will focus to build a model that recognizes a given popular attraction or landmark using Google landmark dataset. This landmark recognition model will be handy to identify the name of a landmark in the image. This will also helpful for photo organization in smartphones and fields like aviation, maps, crime - solving, etc. This Report describes various image mining process followed to build landmark recognition model using google dataset. The report is subdivided into 5 section. Section A – Dataset description, Section B – Image Mining and Feature Extraction process, Section C – Experimental Setup, Section D – Model Results, Section E – Conclusion drawn from the project.

Dataset Description:

For most accurate prediction result and to capture typical appearance of the image, we need large annotated landmark dataset. Google has released its latest landmark dataset named, Google-Landmarks-v2 (September 2019) which makes it our ideal choice for landmark recognition and retrieval purposes. This dataset includes over 5 million images with more than 200,000 diverse landmark classes. Google has published this dataset in 3 sets – train, index and test. The train and test files are used for landmark recognition and index file is used for retrieval purposes. We have used Train set published by google for this project. The major challenge while using this dataset is that of a highly imbalanced training dataset. This is because since there are large number of categories, also many classes with single digit training data which makes it difficult to classify and train the model for such classes. However, the scope of this project is to classify the top 10 sampled image classes from the train set. Top 10 sampled image records were subsided and used for modelling purpose. Then dataset is subdivided into 70-30 ratio as train and test set.

Top 10 Sampled classes:

1. Total Records – 34960
2. Number of classes – 10

Top 10 Landmark IDs –

Landmark_id	Category
192931	York_River_State_Park
177870	Lviv
126637	Corktown,_Toronto
62798	Enchanted_Floral_Gardens_of_Kula
171772	University_of_Chicago_Library
83144	Museum_of_Folk_Architecture_and_Ethnography_in_Pyrohiv
151942	Naturschutzgebiet_Mittleres_Innerstetal_mit_Kanstein
176528	Haleakal National_Park
20409	Noraduz_Cemetery
138982	Media_contributed_by_the_ETH-Bibliothek

1. Dataset is split into 70-30 Ratio. Number of records present each split is given below.
2. Train Set – 24472
3. Test Set - 10488
4. Dataset has three features – id, URL, landmark_id
5. Id – String - unique Id associated with each datapoint
6. URL – String – Describes the image link – mostly in wiki commons
7. Landmark_id – Unique Id to identify each image class – Label class

Individual Work Description:

This portion of the final project describes the part of pre-processing where downloading the images and creating a part of the GUI for the project are implemented.

Data Load & Preprocessing:

Train Dataset from Google landmark dataset is loaded, and top 10 sampled records are identified and stored. This data is used as source data for our project. Dataset is divided two parts – Train & Test sets. From image link given in URL, images are downloaded and saved in two folder – Train_image and Test_image. If image link in dataset is inaccessible or broken, id's associated with data is added to errored is list. As downloaded image are of different dimension, to maintain uniformity, images are resized to aspect ratio – (256,256).

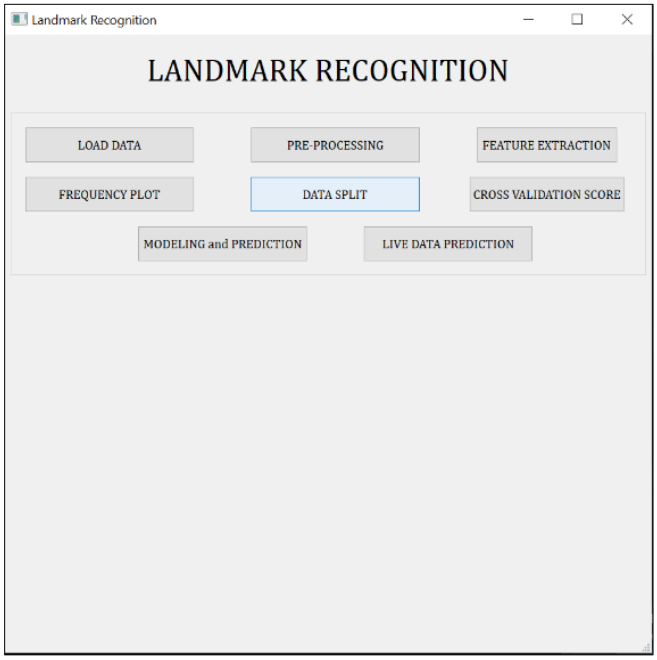
Image_Download.py → This file describes image download and resize process. "download_prep" function is called from main function for every datapoint in train and test data.

After this feature extraction using HOG Classifier is performed. This process is repeated for all images in the dataset, resulting array is saved train feature and test feature list set respectively. Associated labels are saved to test labels and train labels. These are used as input and target variables. To save computational time, as data download and feature extraction for 30k dataset is huge, we have preloaded the data and csv files containing feature and label details are used for analysis purposes.

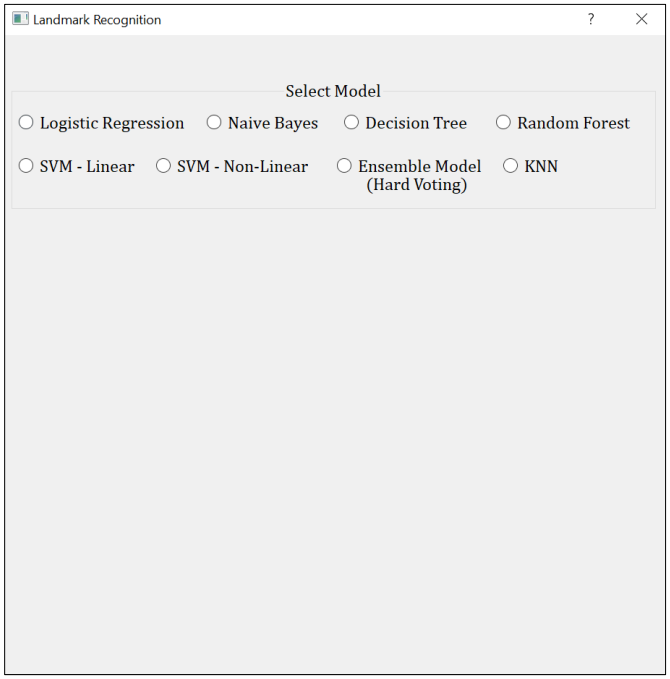
Creating the Interface (GUI):

The GUI was created using PyQt5, which is a cross-platform toolkit that binds to Python. This process involves creating a Main Window, Buttons, Labels and Dialog Boxes. In order to work on the GUI, the PyQt5 package was installed as it works with Python 3 and above. The main class of the GUI, is called the Ui_MainWindow(). This class encompasses all the buttons, labels and any other widgets that we may contain in the main window. Multiple buttons were created for the purpose of displaying the results of a model, when it was clicked. The class Ui_MainWindow() also consists of several functions that allow us to navigate to multiple windows and connecting to functions.

The Main Window consists of several buttons. This portion describes the creation and connection of the Load Data, Pre-Processing and the Modelling and Prediction button. All the buttons are pushButtons. In order to connect the code that loads data to the Load Data button, a new method called load_data() was created which consisted of the Python code for loading the dataset. It was then connected to the button using the clicked.connect() function. Similarly, function preproc() was created and connected to the Pre-Processing button.



The Modelling and Prediction button is connected to a function that maps to a different window, or what is called as a Dialog Box in PyQt5. This dialog box has a main class called Ui_Model(), which consists of the buttons for all the models implemented. This includes buttons for KNN, Logistic Regression, Decision Tree, Random Forest, SVM – Linear, SVM – Non-Linear, Ensemble Model (Hard Voting), and Naïve Bayes. The class Ui_Model() consists of 8 different functions, that contain the codes for each of the models that were implemented. These functions were again connected to the buttons as in the main window. When the button for Modeling and Prediction on the Main Window is click, the following sub – window (Dialog Box) pops up. Here, selecting a model by clicking on the radio button, will display the results of that particular model such as accuracy, classification matrix, etc.



Mywork.py → This file describes the pre-processing action for downloading the images and storing them in multiple folders. It also contains the different codes for the GUI application.

Results:

EDA:
To 10 Sampled Landmark Details

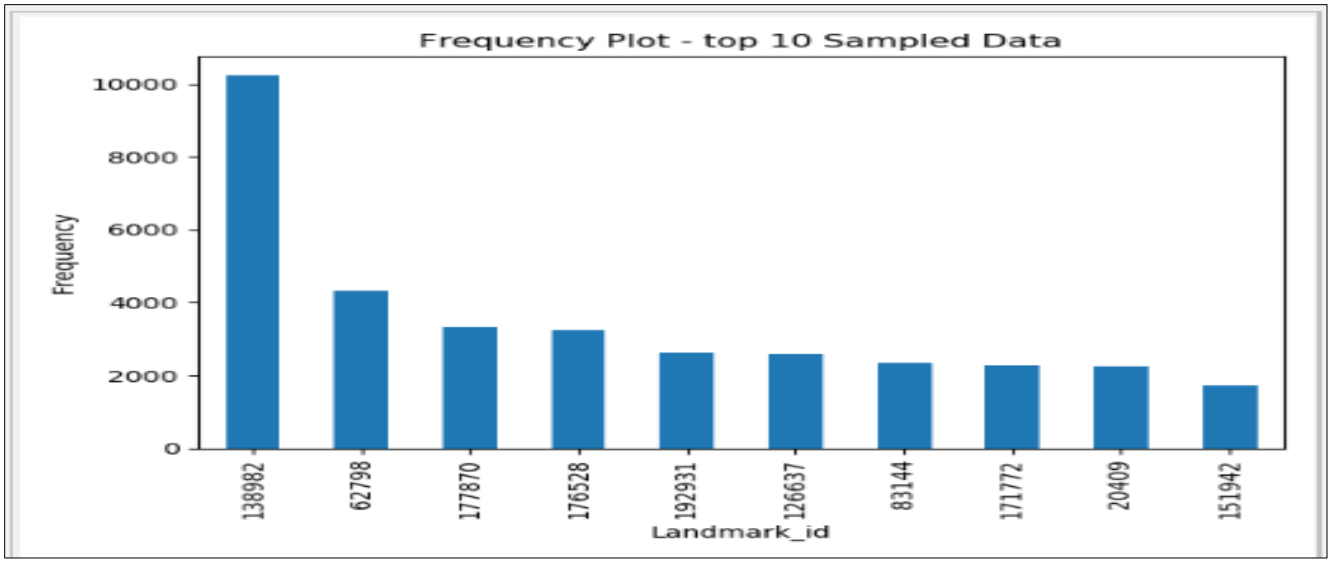


Fig 1: Frequency Plot – top 10 Sampled Data

From Graph, we could see our dataset is highly imbalanced. Landmark id – '138982' has highest annotated images.

Broken or Errored out Links:

```
Below are the errored image ID which is not present in given folder
Missing Files from Train set :  ['89c94048a4ce6f22', 'fb04e87f3db78e9c', '1e3c958352cf8117']
Missing Files from Test set :  ['4405c54c85933767', '292860297de247e5', 'eceea181700f244f', 'e185d73bb5e1b9a9', '1713a87f485ac2d1', 'c1c3663b0abcb15d', '2c840e91a82a2a55']
```

Below findings provides the information on errored out links present in the dataset.

Fig 2: Errored Out Image ID

Summary and Conclusions:

Landmark recognition model is built to classify top 10 sampled landmark id of google dataset. This project explored the possibility of building model with various machine learning algorithm. From comparative study, we could see Random Forest algorithm works best for given dataset followed by logistic Regression. In terms of ensemble model, Random forest with nonlinear SVM gives better classification Model. SVM model doesn't suit for our dataset. As dataset is highly imbalance, it ia hard to find optimum boundary using SVM. Thus, random forest well suited for our landmark recognition data. However, Accuracy achieved is 68%, which is not great. As dataset is huge and imbalance, if we increase class scalability, these algorithms may not work best for recognizing landmark. In such cases we can use neural network may

works better. Also, many classes have least datapoints, if we get more annotated images, prediction percentage may increase further.

Percentage of code from Internet: Apart from pyqt5 & import codes, I have used 70 lines of code, out of which I have used few syntax and statements from net which is 10 lines and modified 5 lines. So total percentage of code used from net is 14%

Reference Materials:

1. Announcing Google-Landmarks-v2: An Improved Dataset for Landmark Recognition & Retrieval (2019, September),
   Retrieved from: https://ai.googleblog.com/2019/05/announcing-google-landmarks-v2-improved.html

2. The Common Visual Data Foundation(2019, September), Google Landmarks Dataset v2,
   Retrieved from: https://www.kaggle.com/c/landmark-recognition-2019

3. Y. Li, D. J. Crandal and D. P. Huttenlocher, Landmark Classification in Large-scale Image Collections,
   Retrieved from: https://www.cs.cornell.edu/~yuli/papers/landmark.pdf

4. A. Crudge, W. Thomas and K. Zhu, Landmark Recognition Using Machine Learning,
   Retrieved from: h ttp://cs229.stanford .edu/proj2014/Andrew%20Crudge, %20Will%20Thomas,%20Kaiyuan%20Zhu,%20Landmark%20Recognition%20Using%20Machine%20Learning.pdf

5. Y. Takeuchi, P. Gros, M. Hebert and K. Ikeuchi, Visual Learning for Landmark Recognition,
   Retrieved from: https://www.cs.cmu.edu/~takeuchi/iuw97/iuw97.html https://www. A n alyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/

6. DelftStack,
   Retrieved from: https://www.delftstack.com/tutorial/pyqt5/pyqt5-label/

7. TutorialsPoint,
   Retrieved from: https://www.tutorialspoint.com/pyqt/pyqt_qlabel_widget.htm

8. Python Tutorials,
   Retrieved from: https://pythonspot.com/pyqt5-image/

9. TechwithTim,
   Retrieved from: https://techwithtim.net/