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 ID –

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

Data Mining Algorithm:

In this project, we have used HOG classifier for feature extraction and created comparative study how dataset reacts to various classifier like Logistic Regression, SVM, Naïve Bayes, KNN, Random Forest, Decision Tree and ensemble - Voting Classifier.

HOG Feature Descriptor:

Feature descriptor captures and represents important information of the images. HOG descriptor is widely used feature extractor in fields of computer vision tasks. It captures both gradient and orientation of the image which makes it unique from other descriptor, like apart by identifying edge pixel, it also gives edge direction. HOG classifier calculates gradients for every pixel in the image store as matrix. The change in x direction and y direction - gradients are measured [Gx & Gy] and stored. Then Magnitude and orientation of each gradients are measured. Frequency table is formed based on orientation & magnitude levels and distributed over the bins. Hence histograms describing given values can be drawn. Image is subdivided many cells, histograms for each subdivided cell is calculated which gives the features representing in that cell. Thereby important high-level information is generated. For this project, we have used HOG classifier from OpenCV package.

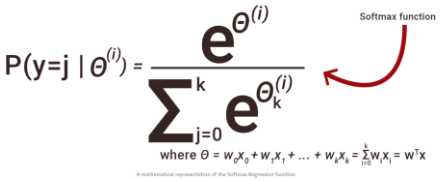
Magnitude: $\sqrt{[(G_x)^2+(G_y)^2]}$

Orientation: $\Phi = \text{atan}(Gy / Gx)$

Modelling:

Logistic Regression:

We have made of use of multiclass logistic regression from sklearn package for our project. Logistic Regression is widely used algorithm for classification problems. For multiclass problems, logistic regressor make use of softmax function. The output are between the range [0,1], input are fed to function which takes ratio of exponential of input to sum of exponential of all input values.



Solver – finds the parameters weights that minimize cost function – mostly used in regression scenario. In our project, we used lbfgs – default – fast for larger dataset.

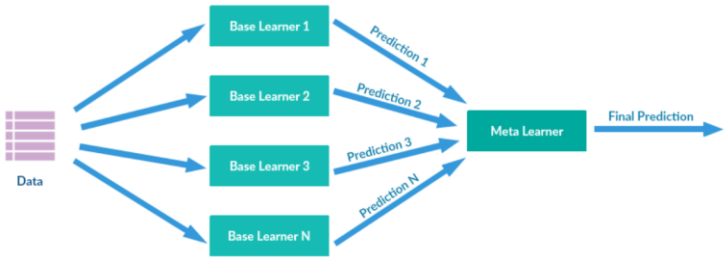
SVM:

Support Vector machine are most popular and commonly used non neural algorithm for image classification, as it gives more optimum decision boundary between classes. There are many Kernel that SVM uses for deciding decision spread. For given dataset, its always crucial task to find apt kernel. We have used both linear and non-linear kernel of SVM module in our project. For Nonlinear SVM kernel, there are two parameters that relatively affects the performance – "c" and "gamma". "C" represents cost/penalty tradeoff for misclassifying the datapoint. C- low – high bias and low variance – misclassification is high and "gamma" – spread of kernel around the data point[decision region] , lower the value, decision boundary are board and there are chances of overfitting the data. To decide the apt parameter, we used grid search on all c values and gamma values from 1 to 10 and best hyper parameter is identified for the modelling purpose.

C values represents

Ensemble Method:

To increase predictive power, ensemble methods are recommended, where results of various model is fed to give more robust results. We used Voting classifier which takes results from other algorithms and result that has majority vote is used as final prediction.



Stacking visualized; Image from http://supunsetunga.blogspot.com/

In this project, we used seven base algorithms and predicted the results. These outputs are given as input to voting classifier and final prediction is determined. Prediction score may increase or decrease based on results you stack up to the voting classifier. To get best results, its necessary to try various combinations of result sets. Hence, we have tried all combination of input sets to the voting classifier and best stacking estimators are identified for final ensemble prediction.

We have also used Random forest with 500 trees, decision tree, Naïve Bayes, KNN (5 Neighbors) algorithms in this project and predicted the results.

In order to access model performance for varying samples, cross-validation [10 Fold] is performed om train set and cross validation score is calculated for each model.

Experimental Setup:

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.

Once Image is processed, we have used HOG classifier for feature extraction from loaded images. **Feature_Extraction.py** – This file contains "hog" function – which calculates the gradients and orientation for each pixel values in image and histogram is derived for each cell. The feature details are then stacked as Numpy array and final Numpy array contains feature of the image is returned.

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.

Modeling:

As we have one input feature variable, we couldn't able to hyper tune the parameters with respect to variables. We experimented with various model parameters that best fit for our dataset.

Model_Function.py – This File contains model functions used for this project. It takes the train set feature and labels, fit the model and returns the predicted label set.

Logistic Regression:

Method - LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomial',max_iter = 1000)

Package: Sklearn

Parameters: multi_class = “multinomial” – To support multiple class as we have 10 class variables
Solver = “lbfgs” – It is fast for last dataset & saves memory.

SVM: Choice of Kernel for SVM is problem dependent, we have used both linear kernel and ‘rbf’ kernel and checks the model performance which is ideal for our dataset
Package: Sklearn

a) Linear Model:
   Method: SVC (kernel='linear', max_iter=1000)
   Parameter: Kernel = ‘Linear’ – Try to fit linear line between classes.
b) Non-Linear Model:
   Method: SVC (kernel="rbf", max_iter = 1000)
   Parameter Values: We used Grid search to determine C and gamma values for Non-linear model
   Grid Search: params_dict = {"C": np.logspace(-1, 3, 10), "gamma": np.linspace(0.0001, 10, 10)}
   svm = SVC (kernel="rbf", max_iter = 1000)
   GridSearchCV(estimator=svm, param_grid=params_dict)

   Best estimators from result are fed to SVM model for training purpose.
   Best Estimators identified by grid search is C = 0.1, gamma = 0.0001

Random Forest:
Method - RandomForestClassifier(n_estimators=500)
Package: Sklearn
Parameters: n_Estimators = 500 – Number of trees used – 500.

Decision Tree:
Method - DecisionTreeClassifier(random_state=seed, criterion = ‘gini’)
Package: Sklearn
Parameters: criterion = ‘gini’. All other default parameters are used - min_samples_leaf=1, min_samples_split=2, splitter='best'

KNN:
Method - KNeighborsClassifier()
Package: Sklearn
Parameters: metric='minkowski', n_neighbors=5

Naïve Bayes:
Method - GaussianNB()
Package: Sklearn
Parameters: Default parameters

Best_voting – This function uses various combination results from all the above algorithm and get best predicted result.

All the combinations of 7 algorithm is fitted to model and accuracy of the predicted result is measured. Combination which produces best accuracy is used as best combination and used for ensemble method.

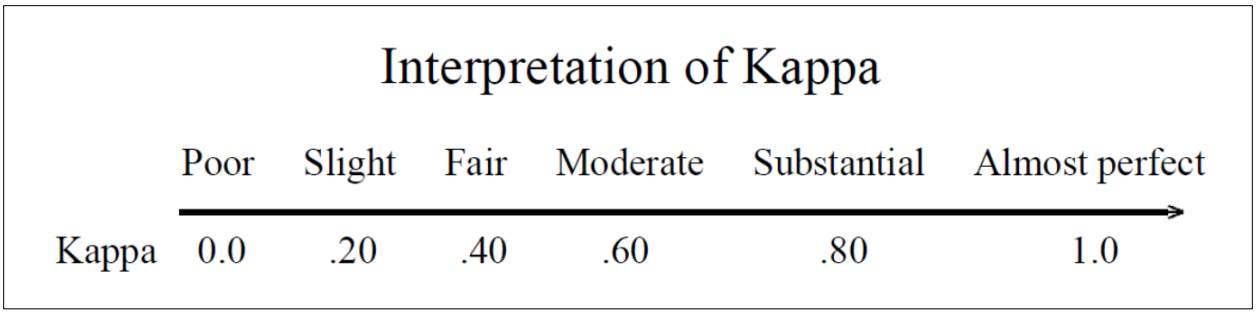
VotingClassifier from ensemble class of sklearn package is used.

Cross Validation Score:

Apart from train and test set evaluation, we have used 10-fold cross validation to reconfirm model performance for varying sample set of test data. cross_val_score method from model selection package is used for this process.

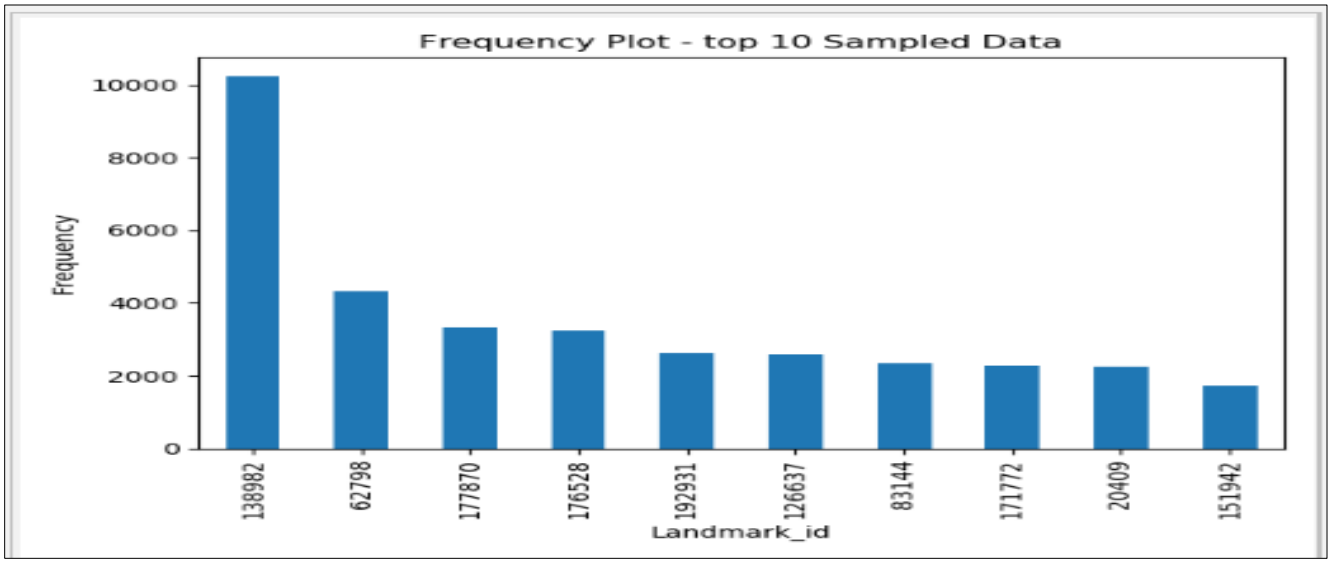
Model Evaluation:

1. Accuracy Score – Predicted values is compared with actual label values, accuracy of the prediction is calculated. This gives information number of data points classified properly in our dataset.
2. Confusion Matrix – Matrix gives information on all 10 classes predicted and actual value statistics, from which we could able to conclude on class that were classified wrongly more often.
3. Classification Report – Precision, recall gives the percentage of prediction rate for each class.
4. Cohen Kappa score – As our dataset is imbalanced, Cohen kappa score gives the information each class in multi class environment



Results:

EDA:



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

```
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']
```

Fig 2: Errored Out Image ID

CONFUSION MATRIX:
Below Shows the Confusion Matrix obtained for the different models.

Model Evaluation Metrics - Logistic Regression

Confusion Matrix	126637.0	138982.0	151942.0	171772.0	176528.0	177870.0
126637.0	378	77	20	37	14	1
138982.0	17	922	39	3	31	28
151942.0	19	119	201	38	32	19
171772.0	29	19	35	430	23	2
176528.0	19	15	4	16	2954	0
177870.0	17	190	43	9	14	41
192931.0	6	72	6	50	56	3
20409.0	70	267	20	27	27	7
62798.0	42	38	62	130	57	12
83144.0	39	170	70	44	34	17

	192931.0	20409.0	62798.0	83144.0
126637.0	6	34	58	32
138982.0	35	136	22	50
151942.0	16	103	97	62
171772.0	34	13	153	36
176528.0	11	11	34	10
177870.0	18	108	35	42
192931.0	374	43	38	14
20409.0	42	447	37	23
62798.0	44	51	513	85
83144.0	33	42	139	211

Model Evaluation Metrics - Random Forest

Confusion Matrix	126637.0	138982.0	151942.0	171772.0	176528.0	177870
126637.0	396	77	19	29	5	1
138982.0	2	1140	12	1	4	14
151942.0	30	71	254	40	3	16
171772.0	15	22	7	483	6	0
176528.0	4	22	3	14	2942	1
177870.0	26	187	65	8	2	89
192931.0	0	21	1	20	13	1
20409.0	20	291	22	30	8	16
62798.0	43	70	66	122	13	13
83144.0	20	152	79	41	6	14

	192931.0	20409.0	62798.0	83144.0
126637.0	6	43	39	42
138982.0	23	41	9	37
151942.0	28	64	106	94
171772.0	37	15	154	35
176528.0	18	18	45	7
177870.0	17	54	29	40
192931.0	565	3	17	21
20409.0	41	492	16	31
62798.0	38	49	513	107
83144.0	35	38	135	279

Fig 3: Confusion Matrix – Logistic regression & Random Forest

Model Evaluation Metrics - SVM Linear Model

Confusion Matrix	126637.0	138982.0	151942.0	171772.0	176528.0	177870.
126637.0	16	9	13	35	53	23
138982.0	50	211	27	25	351	46
151942.0	7	35	21	51	41	17
171772.0	10	13	65	191	43	30
176528.0	29	14	82	969	1237	4
177870.0	7	67	21	25	50	26
192931.0	2	4	15	175	9	3
20409.0	15	81	18	50	129	40
62798.0	26	29	70	101	103	34
83144.0	12	50	17	73	92	31

	192931.0	20409.0	62798.0	83144.0
126637.0	3	233	159	113
138982.0	22	212	154	185
151942.0	3	236	162	133
171772.0	4	120	211	87
176528.0	21	436	156	126
177870.0	6	137	86	92
192931.0	7	357	64	26
20409.0	5	451	84	94
62798.0	9	177	305	180
83144.0	4	168	186	166

Model Evaluation Metrics - SVM Non Linear Model

Confusion Matrix	126637.0	138982.0	151942.0	171772.0	176528.0	177870.0
126637.0	0	0	0	0	0	0
138982.0	0	157	0	0	0	0
151942.0	0	0	0	0	0	0
171772.0	0	0	0	0	0	0
176528.0	0	0	0	0	0	0
177870.0	0	0	0	0	0	0
192931.0	0	0	0	0	0	0
20409.0	0	0	0	0	0	0
62798.0	0	0	0	0	0	0
83144.0	0	0	0	0	0	0

	192931.0	20409.0	62798.0	83144.0
126637.0	0	0	0	657
138982.0	0	0	0	1126
151942.0	0	0	0	706
171772.0	0	0	0	774
176528.0	0	0	0	3074
177870.0	0	0	0	517
192931.0	0	0	0	662
20409.0	0	3	0	964
62798.0	0	0	0	1034
83144.0	0	0	0	799

Fig 4: Confusion Matrix – SVM Model

Model Evaluation Metrics - Decision Tree

Confusion Matrix	126637.0	138982.0	151942.0	171772.0	176528.0	177870.0
126637.0	283	42	44	47	10	28
138982.0	32	789	38	14	10	89
151942.0	52	50	154	46	18	68
171772.0	53	14	55	284	23	22
176528.0	18	27	21	34	2859	15
177870.0	24	113	58	15	6	96
192931.0	9	22	23	40	20	17
20409.0	59	183	62	50	15	79
62798.0	60	54	102	145	34	64
83144.0	38	80	90	63	23	79

	192931.0	20409.0	62798.0	83144.0
126637.0	10	68	79	46
138982.0	19	153	50	89
151942.0	23	68	128	99
171772.0	36	31	172	84
176528.0	16	19	40	25
177870.0	12	75	59	59
192931.0	426	33	42	30
20409.0	42	350	59	68
62798.0	44	60	313	158
83144.0	41	63	138	184

Model Evaluation Metrics - KNN

Confusion Matrix	126637.0	138982.0	151942.0	171772.0	176528.0	177870
126637.0	406	59	56	28	10	9
138982.0	45	922	29	3	5	70
151942.0	127	84	203	40	38	35
171772.0	80	21	34	407	98	3
176528.0	39	27	29	39	2863	3
177870.0	27	190	66	4	12	86
192931.0	10	42	7	57	89	4
20409.0	67	300	53	16	20	69
62798.0	155	43	135	157	97	33
83144.0	130	108	102	50	66	45

	192931.0	20409.0	62798.0	83144.0
126637.0	7	24	29	29
138982.0	40	121	10	38
151942.0	13	36	67	63
171772.0	22	11	67	31
176528.0	15	12	32	15
177870.0	30	57	11	34
192931.0	404	25	12	12
20409.0	76	328	12	26
62798.0	21	31	274	88
83144.0	25	37	75	161

Fig 5: Confusion Matrix – Decision Tree & KNN

Model Evaluation Metrics - Naive Bayes

Confusion Matrix	126637.0	138982.0	151942.0	171772.0	176528.0	177870.0
126637.0	301	52	29	14	185	7
138982.0	113	695	104	18	47	76
151942.0	165	102	118	58	97	36
171772.0	111	32	22	229	195	15
176528.0	37	31	5	17	2911	3
177870.0	110	175	39	19	56	39
192931.0	36	10	22	26	291	2
20409.0	282	207	44	39	183	24
62798.0	228	134	86	122	142	46
83144.0	138	148	96	76	70	58

	192931.0	20409.0	62798.0	83144.0
126637.0	14	14	14	27
138982.0	29	79	15	107
151942.0	51	8	22	49
171772.0	81	4	46	39
176528.0	29	6	29	6
177870.0	24	13	7	35
192931.0	247	6	11	11
20409.0	72	69	20	27
62798.0	71	22	81	102
83144.0	46	14	33	120

Model Evaluation Metrics - Ensemble

Accuracy Score : 67.5833094624272

Confusion Matrix	126637.0	138982.0	151942.0	171772.0	176528.0	177870.
126637.0	393	77	29	34	5	2
138982.0	8	1123	21	2	3	19
151942.0	31	77	257	40	3	14
171772.0	19	23	11	486	4	0
176528.0	5	23	6	15	2941	0
177870.0	23	189	68	8	4	88
192931.0	0	19	4	22	13	2
20409.0	23	277	28	30	9	16
62798.0	47	68	78	138	14	8
83144.0	21	150	81	44	9	16

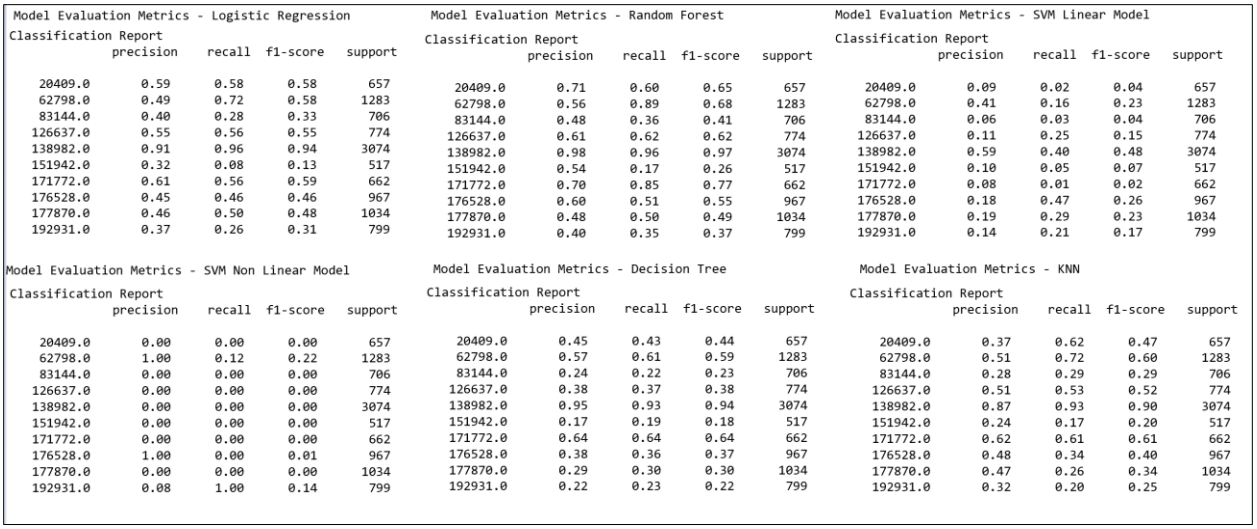
	192931.0	20409.0	62798.0	83144.0
126637.0	6	40	37	34
138982.0	17	42	16	32
151942.0	29	64	109	82
171772.0	34	14	146	37
176528.0	16	20	40	8
177870.0	21	50	30	36
192931.0	567	2	17	16
20409.0	46	484	25	29
62798.0	39	41	474	127
83144.0	42	37	134	265

Fig 6: Confusion Matrix – Naïve Bayes & Ensemble Model

From Confusion Matrix, we could see landmark id – 177870,151942 has more misclassified records. SVM Model doesn't perform well in predicting classes in the given dataset. We can infer that landmark id – 177870,151942 has similar feature with other classes which makes it hard to distinguish from others.

CLASSIFICATION REPORT:

Fig 7: Classification report



Model Evaluation Metrics - Logistic Regression

Classification Report

	precision	recall	f1-score	support
20409.0	0.59	0.58	0.58	657
62798.0	0.49	0.72	0.58	1283
83144.0	0.40	0.28	0.33	706
126637.0	0.55	0.56	0.55	774
138982.0	0.91	0.96	0.94	3074
151942.0	0.32	0.08	0.13	517
171772.0	0.61	0.56	0.59	662
176528.0	0.45	0.46	0.46	967
177870.0	0.46	0.50	0.48	1034
192931.0	0.37	0.26	0.31	799

Model Evaluation Metrics - Random Forest

Classification Report

	precision	recall	f1-score	support
20409.0	0.71	0.60	0.65	657
62798.0	0.56	0.89	0.68	1283
83144.0	0.48	0.36	0.41	706
126637.0	0.61	0.62	0.62	774
138982.0	0.98	0.96	0.97	3074
151942.0	0.54	0.17	0.26	517
171772.0	0.70	0.85	0.77	662
176528.0	0.60	0.51	0.55	967
177870.0	0.48	0.50	0.49	1034
192931.0	0.40	0.35	0.37	799

Model Evaluation Metrics - SVM Linear Model

Classification Report

	precision	recall	f1-score	support
20409.0	0.09	0.02	0.04	657
62798.0	0.41	0.16	0.23	1283
83144.0	0.06	0.03	0.04	706
126637.0	0.11	0.25	0.15	774
138982.0	0.59	0.40	0.48	3074
151942.0	0.10	0.05	0.07	517
171772.0	0.08	0.01	0.02	662
176528.0	0.18	0.47	0.26	967
177870.0	0.19	0.29	0.23	1034
192931.0	0.14	0.21	0.17	799

Model Evaluation Metrics - SVM Non Linear Model

Classification Report

	precision	recall	f1-score	support
20409.0	0.00	0.00	0.00	657
62798.0	1.00	0.12	0.22	1283
83144.0	0.00	0.00	0.00	706
126637.0	0.00	0.00	0.00	774
138982.0	0.00	0.00	0.00	3074
151942.0	0.00	0.00	0.00	517
171772.0	0.00	0.00	0.00	662
176528.0	1.00	0.00	0.01	967
177870.0	0.00	0.00	0.00	1034
192931.0	0.08	1.00	0.14	799

Model Evaluation Metrics - Decision Tree

Classification Report

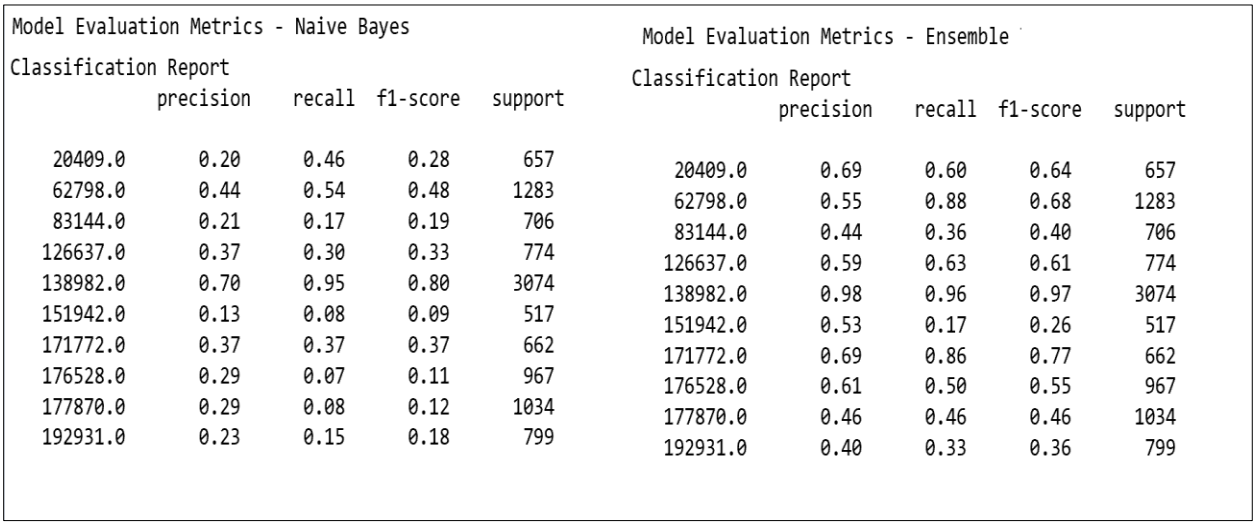
	precision	recall	f1-score	support
20409.0	0.45	0.43	0.44	657
62798.0	0.57	0.61	0.59	1283
83144.0	0.24	0.22	0.23	706
126637.0	0.38	0.37	0.38	774
138982.0	0.95	0.93	0.94	3074
151942.0	0.17	0.19	0.18	517
171772.0	0.64	0.64	0.64	662
176528.0	0.38	0.36	0.37	967
177870.0	0.29	0.30	0.30	1034
192931.0	0.22	0.23	0.22	799

Model Evaluation Metrics - KNN

Classification Report

	precision	recall	f1-score	support
20409.0	0.37	0.62	0.47	657
62798.0	0.51	0.72	0.60	1283
83144.0	0.28	0.29	0.29	706
126637.0	0.51	0.53	0.52	774
138982.0	0.87	0.93	0.90	3074
151942.0	0.24	0.17	0.20	517
171772.0	0.62	0.61	0.61	662
176528.0	0.48	0.34	0.40	967
177870.0	0.47	0.26	0.34	1034
192931.0	0.32	0.20	0.25	799

Fig 7a: Classification report



Model Evaluation Metrics - Naive Bayes

Classification Report

	precision	recall	f1-score	support
20409.0	0.20	0.46	0.28	657
62798.0	0.44	0.54	0.48	1283
83144.0	0.21	0.17	0.19	706
126637.0	0.37	0.30	0.33	774
138982.0	0.70	0.95	0.80	3074
151942.0	0.13	0.08	0.09	517
171772.0	0.37	0.37	0.37	662
176528.0	0.29	0.07	0.11	967
177870.0	0.29	0.08	0.12	1034
192931.0	0.23	0.15	0.18	799

Model Evaluation Metrics - Ensemble

Classification Report

	precision	recall	f1-score	support
20409.0	0.69	0.60	0.64	657
62798.0	0.55	0.88	0.68	1283
83144.0	0.44	0.36	0.40	706
126637.0	0.59	0.63	0.61	774
138982.0	0.98	0.96	0.97	3074
151942.0	0.53	0.17	0.26	517
171772.0	0.69	0.86	0.77	662
176528.0	0.61	0.50	0.55	967
177870.0	0.46	0.46	0.46	1034
192931.0	0.40	0.33	0.36	799

Apart from SVM model results, all other algorithms F1 score is almost similar. Landmark id – 138982 is classified properly most of the times, as we know id – “138982” has 10k samples, more the samples better the classification. Precision and Recall values also follow similar pattern as F1 score. Landmark id – 192931 is misclassified quiet often. Due to imbalance in dataset and multiclass environment, we couldn’t able to conclude much from these results for other classes.

ACCURACY & COHEN KAPPA SCORE:

	Model	Accuracy_Score	Cohen Kappa Score
0	Logistic Regression	61.787453	0.548704
1	Random Forest	68.299437	0.627768
2	SVM_Linear	25.121742	0.141461
3	SVM Non Linear	9.156880	0.015796
4	Decision Tree	54.788504	0.471286
5	KNN	57.805786	0.503012
6	Naive Bayes	45.927623	0.351123
7	Ensemble(Hard voting)	67.583309	0.619471

Fig 8: Accuracy & Cohen Kappa Score

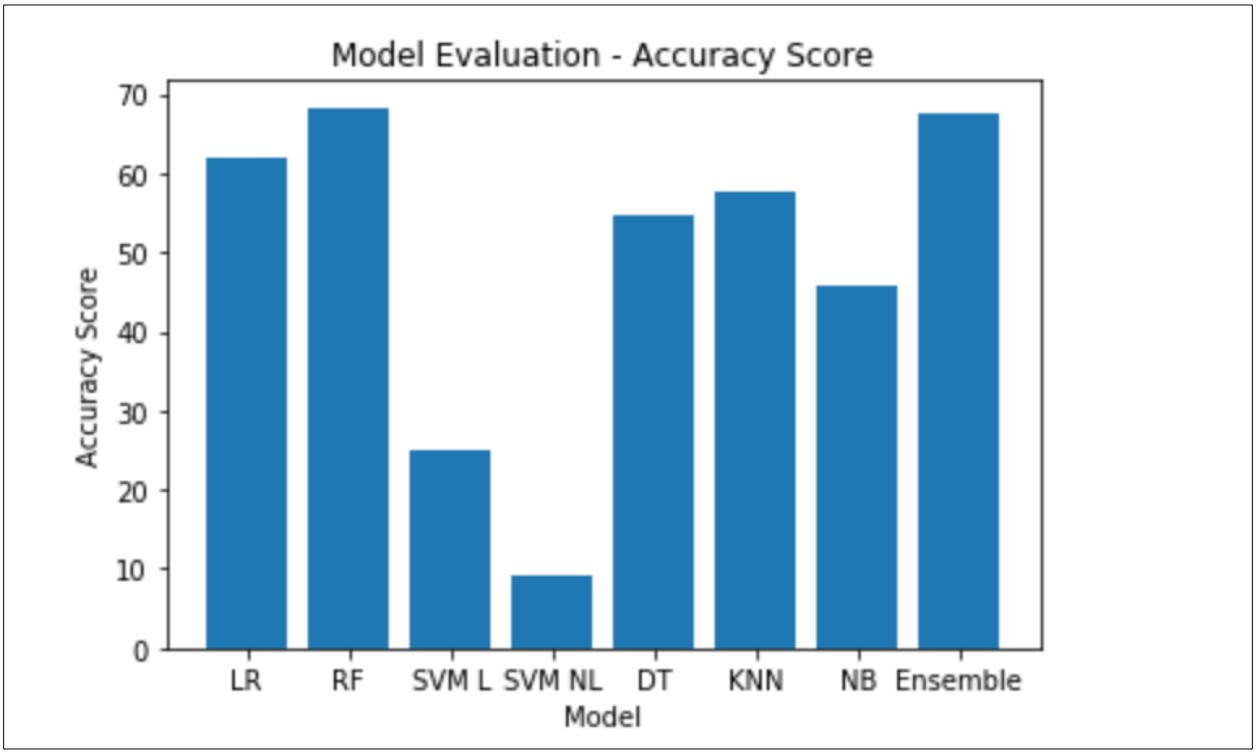


Fig 9: Accuracy Score for various Models

From Accuracy Score and Kappa Score, we could say Random forest gives better accuracy rate and kappa value also falls under Fair agreement region, followed by Ensemble and Logistic Regression. SVM model has lowest accuracy and kappa score, hence, it doesn't suit for given dataset.

CROSS VALIDATION SCORE:
We reconfirm our result, we performed 10-fold cross-validation on trained set. Please find below result for the models.

Model CV Score

	Classifier_Name	Cross_validation_Score
0	LR	0.617887
1	KNN	0.582703
2	Decision Tree	0.552306
3	RF	0.675163
4	NB	0.463691
5	SVM Non Linear	0.090619
6	SVM Linear	0.227345

Fig10: Cross Validation Score

The cross-validation score is similar, we have Random forest, logistic regression with better score and SVM models has least value.

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, its 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.

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: http://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 nalyticsvidhya.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/

10. HOG Classifier Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor
Retrieved from:https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/