# **DATS-6103: Data Mining Final Project Individual 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:**

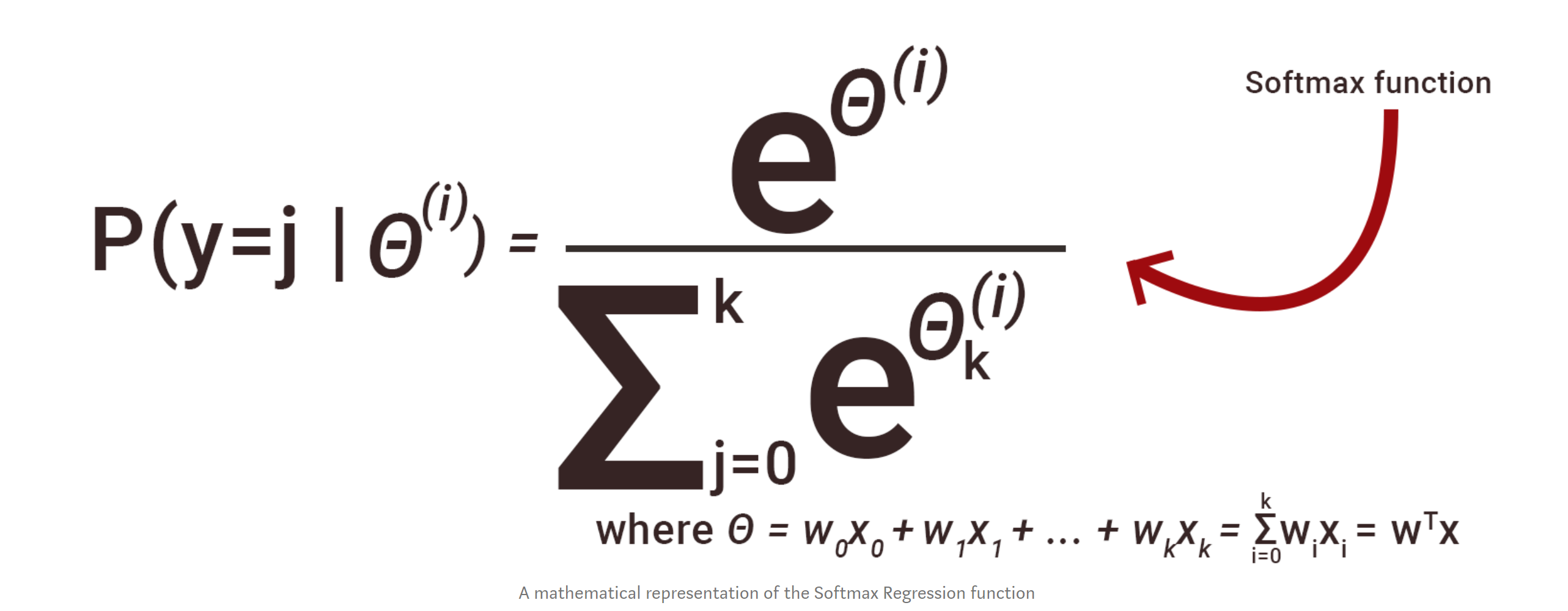
Shared Work: In this project, we have downloaded image from link present in train set, preprocessing like validating the image URL and resizing the image were performed. Then we used HOG feature extractor to extract the each of the information and data is stored as numpy array. These arrays are used as input values and labels associated with these images are used as target class.

Individual Work: Planned to study a comparative analysis various model performance for given dataset and best algorithm suited for our dataset is identified. I have used eight algorithms for this process like logistic regression, Random forest, SVM, KNN, Naïve Bayes, Decision tree and ensemble stacking model. As we have single input variable , we doesn’t have much of the feature tuning to optimize the model, however we have tuned various parameters associated with models to check model performance.

**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.

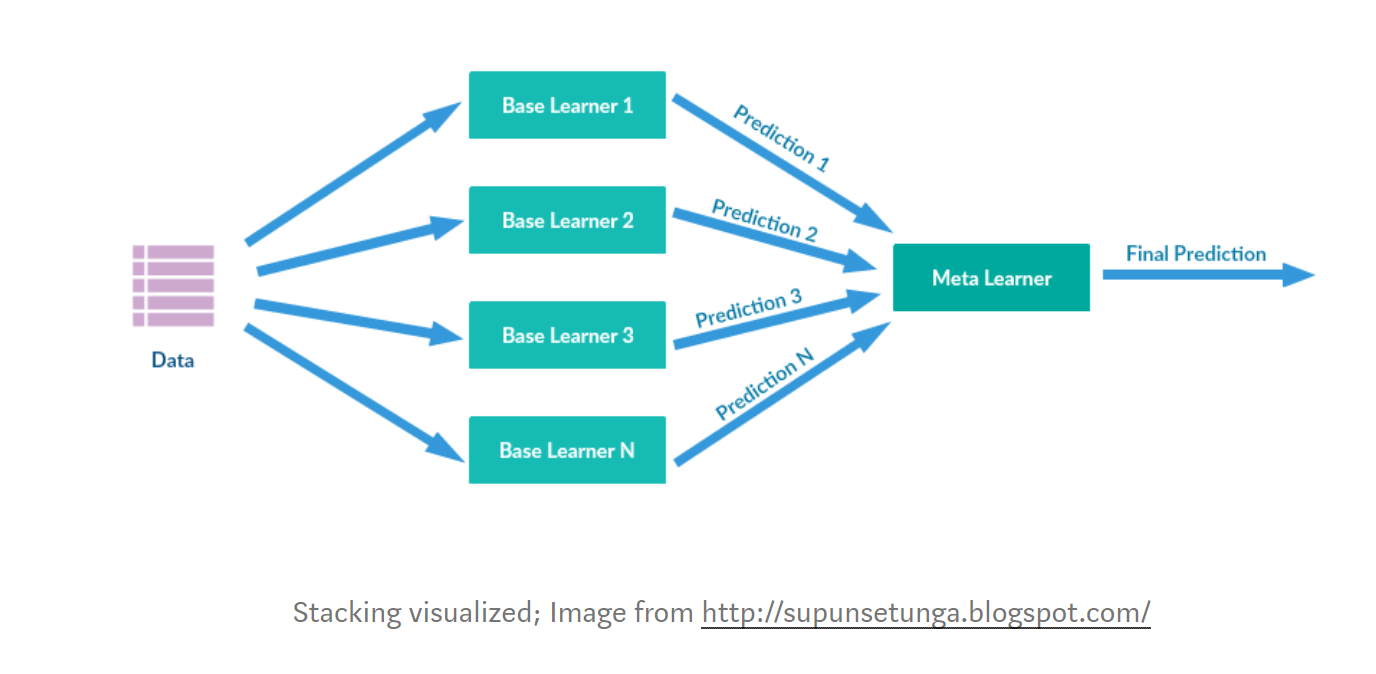
SVM:

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.



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:**

Once data is loaded and saved, dataset is fitted to model and trained. As downloading the image and feature extraction from 30k plus image dataset cost more time, we have saved the data and extracted features already and just loaded the numpy array for our analysis. The Actual code is commented in the respective .py file for reference.

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

1. Linear Model:

Method: SVC (kernel='linear', max\_iter=1000)

Parameter: Kernel = ‘Linear’ – Try to fit linear line between classes.

1. 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.

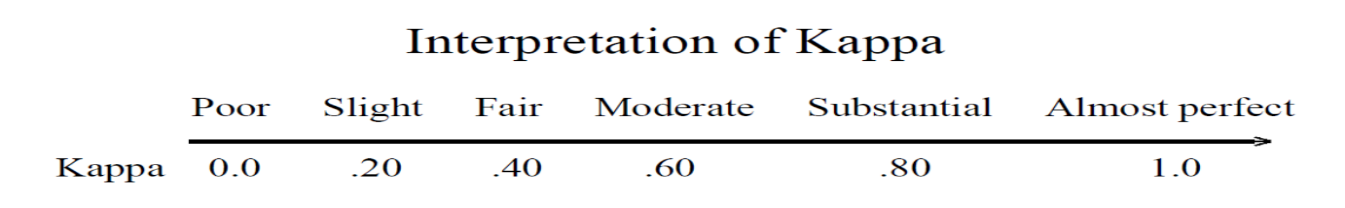
**VotingClassifier** from ensemble class of sklearn package is used.

**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.

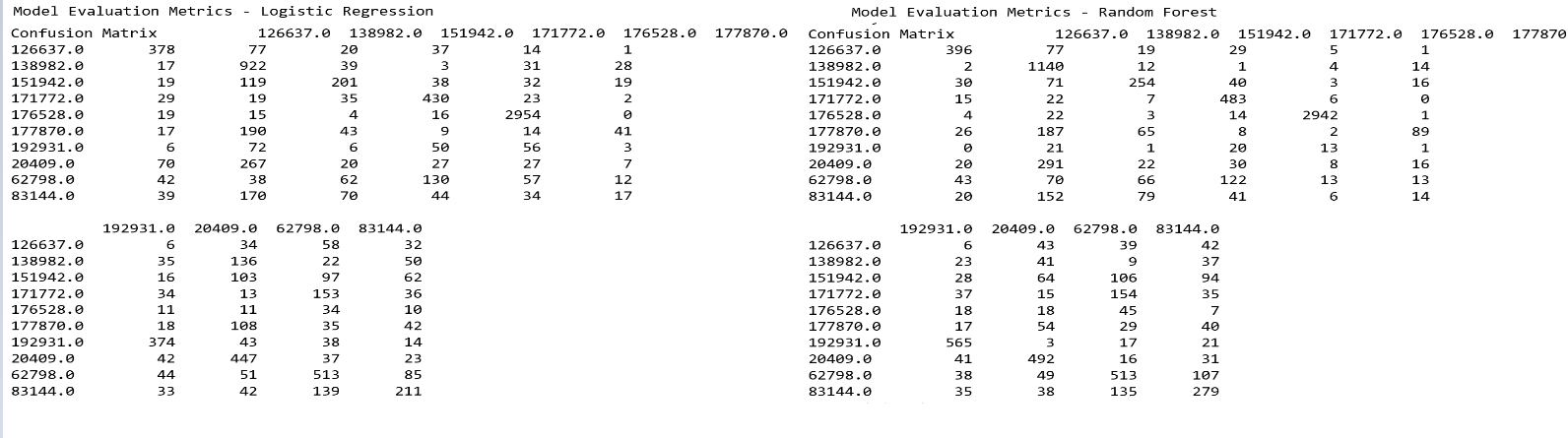
**Model Evaluation:**

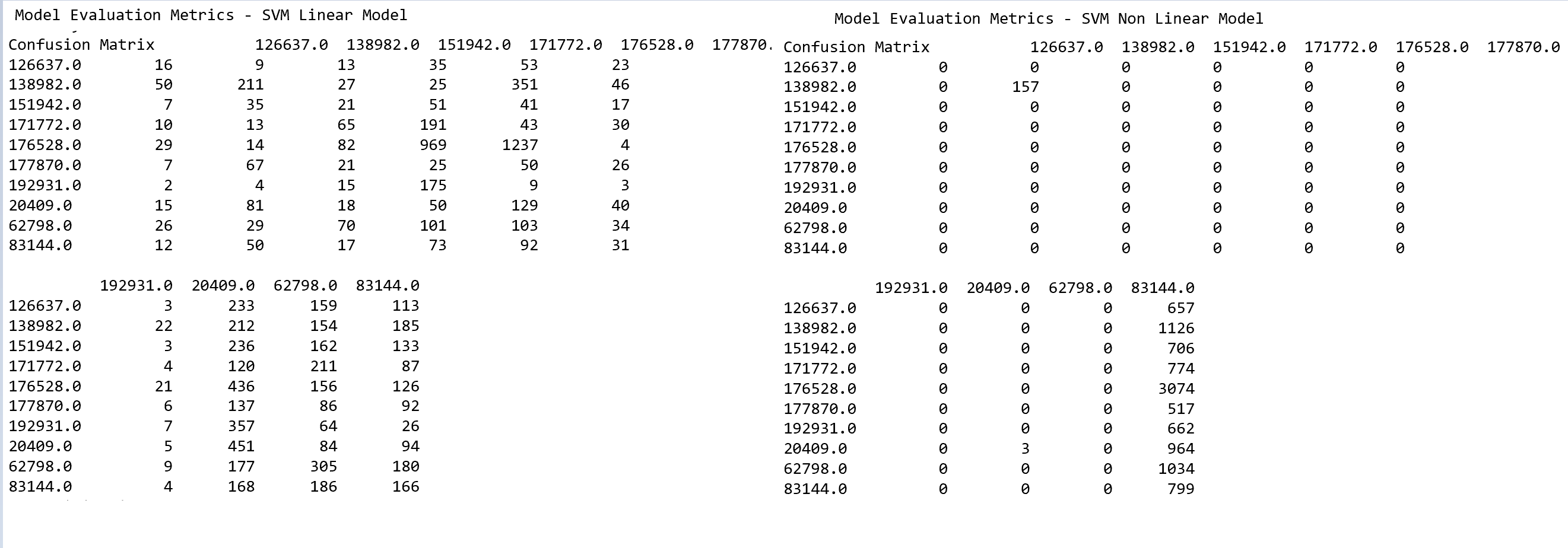
1. Accuracy Score – Predicted values is compared with actual label values, accuracy of the prediction is calculated. This gives information number of data points classified properly in our dataset.
2. Confusion Matrix – Matrix gives information on all 10 classes predicted and actual value statistics, 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:**

**CONFUSION MATRIX:**

Below Shows the Confusion Matrix obtained for the different models.

 Fig 3: Confusion Matrix – Logistic regression & Random Forest

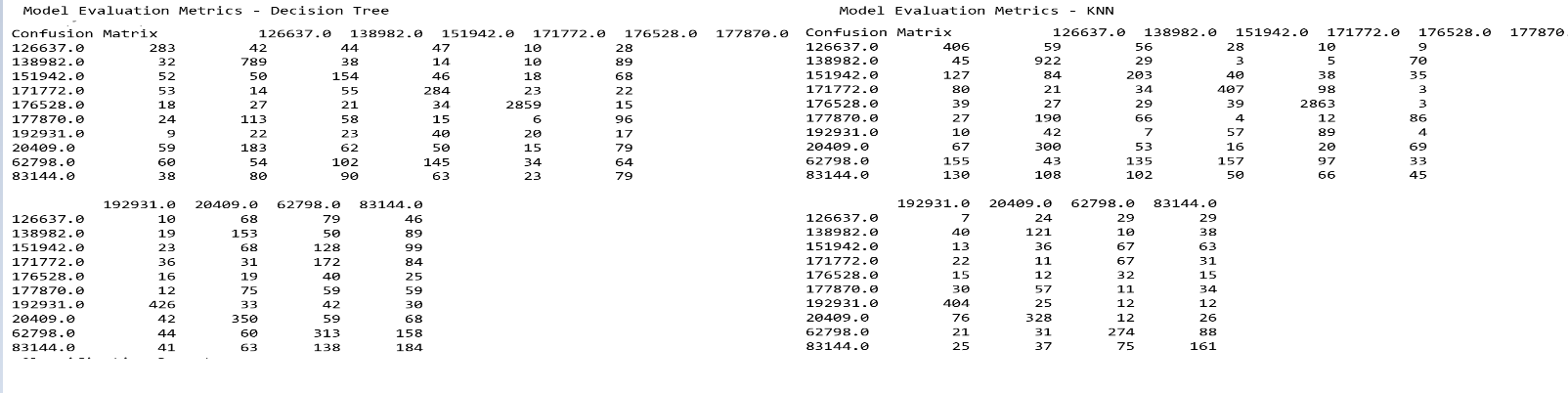
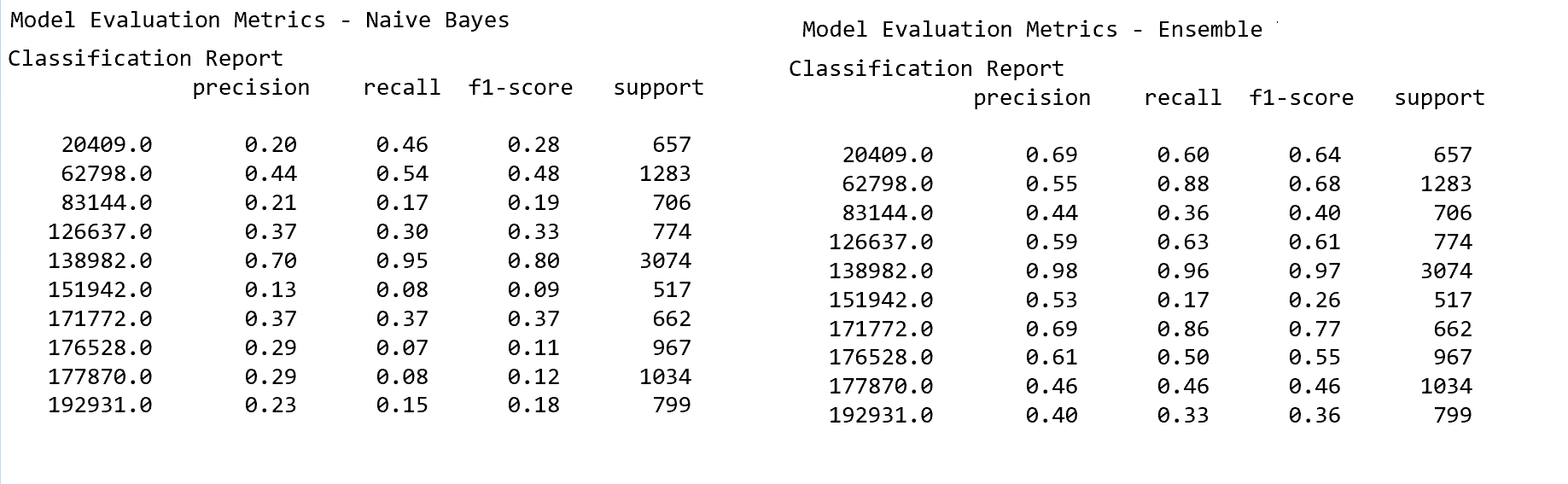
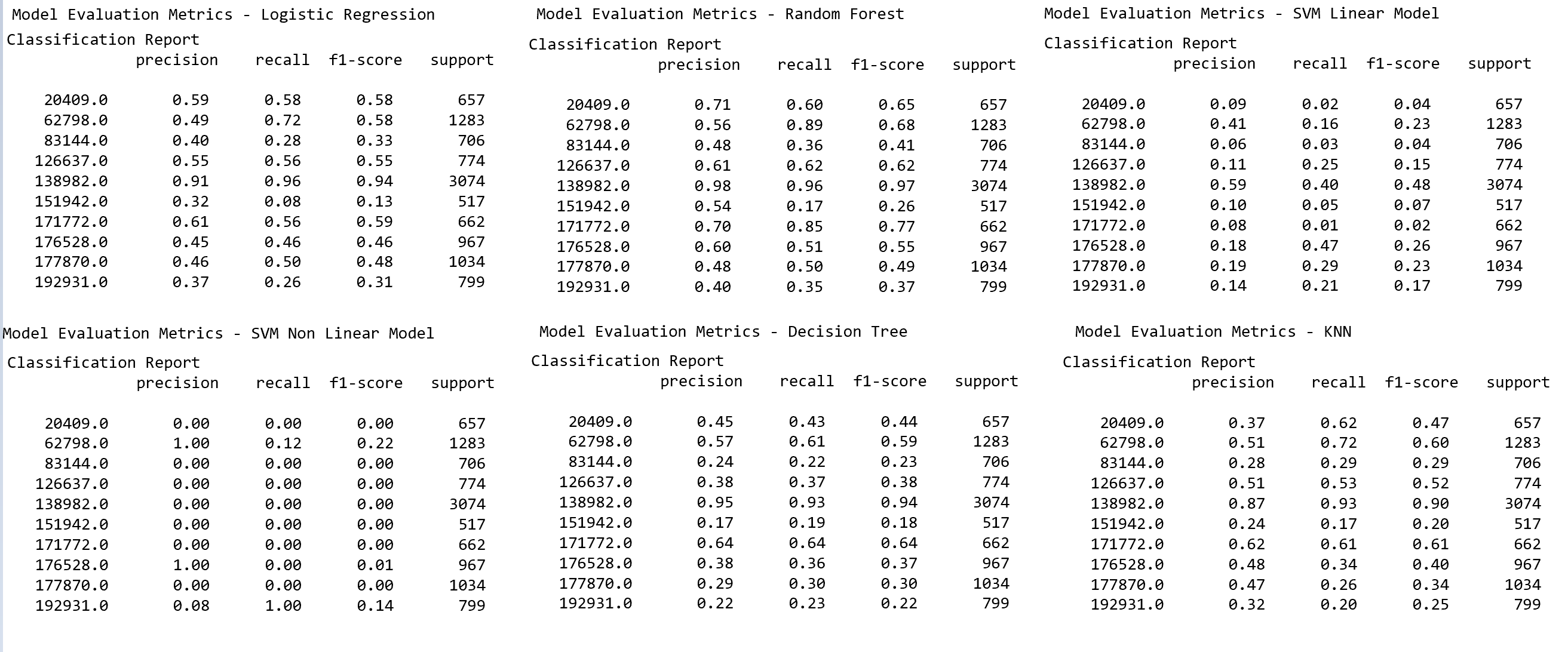
Fig 4: Confusion Matrix – SVM Model

Fig 5: Confusion Matrix – Decision Tree & KNN

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:**

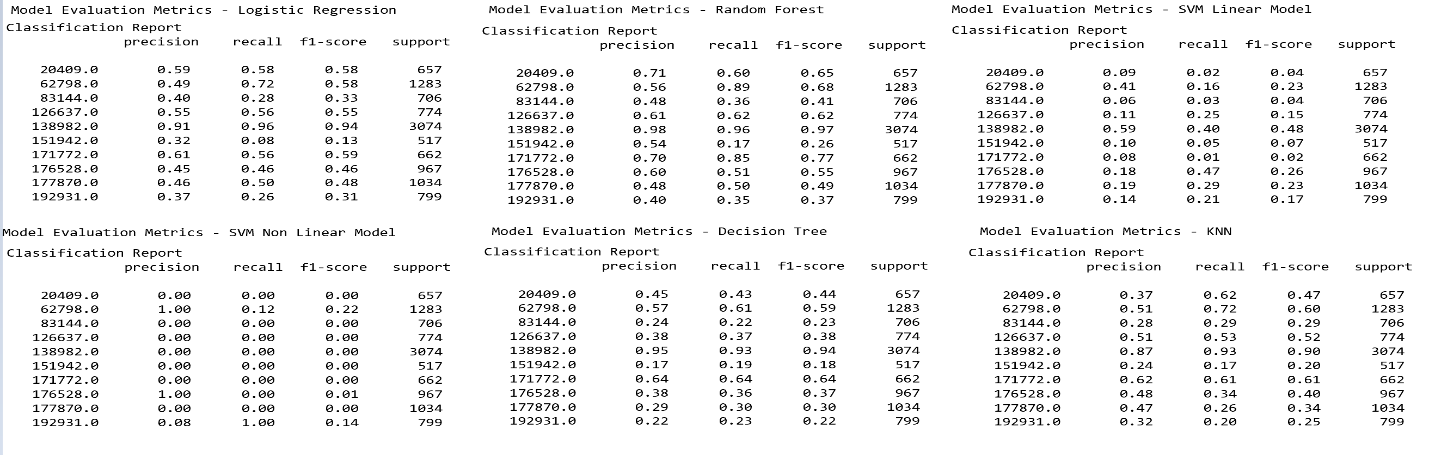
Fig 7: Classification report

Fig 7a: Classification report

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:**

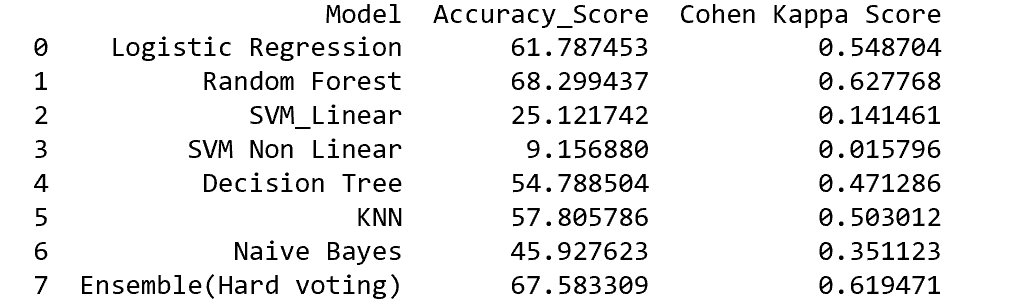


Fig 8: Accuracy & Cohen Kappa Score

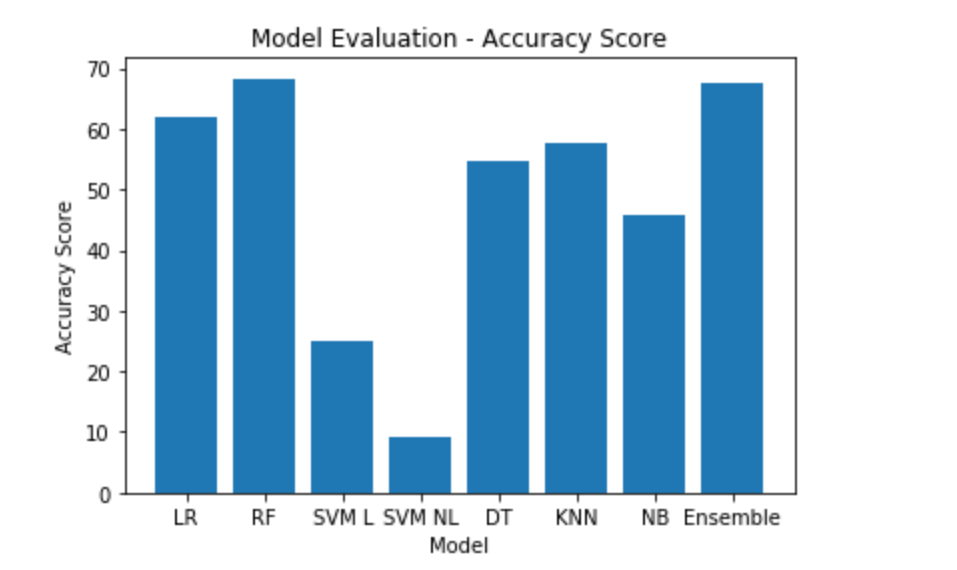


Fig 9: Accuracy Score for various Models

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.

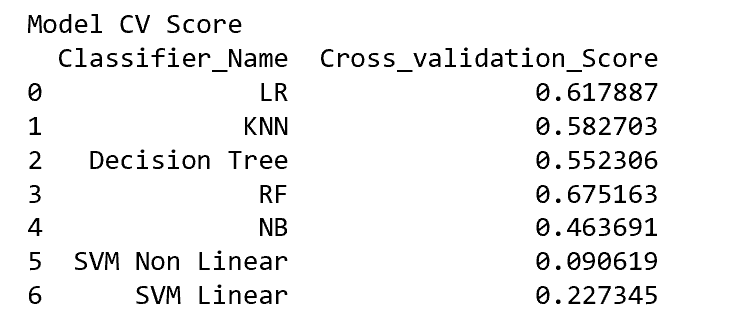


Fig10: Cross Validation Score

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.

Percentage of Code: We couldn’t find any code for Landmark recognition with non-neural network, we have mostly used syntax and aligned the logic to our problem set. Total Lines in code apart from Pyqt5 & import functions are 300 lines approximates, on which modified lines are 100 and used lines are 50 , so 16% of code were copied from internet in modelling.

**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>

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

1. 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

1. 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/

1. DelftStack,

**Retrieved from:** <https://www.delftstack.com/tutorial/pyqt5/pyqt5-label/>

1. TutorialsPoint,

**Retrieved from:** <https://www.tutorialspoint.com/pyqt/pyqt_qlabel_widget.htm>

1. Python Tutorials,

**Retrieved from:** <https://pythonspot.com/pyqt5-image/>

1. TechwithTim,

**Retrieved from:** <https://techwithtim.net/>

1. 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/