Program 2 – Image Classification

Poojitha Amin
SJSU ID: 011811306

Rank – 2
F1-Score - 0.8464

1. Approach/Techniques Used

Dimensionality Reduction - Feature Selection

In order to reduce the number of dimensions used in the model development, a subset of the relevant features was selected for model construction. The method includes and excludes features from the dataset without changing them. It mutes out features that are not so useful and are redundant or irrelevant to reduce the complexity of the model and in turn achieves better accuracy. I have made use of feature_selection module from the Scikit Learn library.

- SelectKBest removes all but the k highest scoring features.

Classification - ExtraTreeClassifier

Trained the model with multiple classifiers like the Naive Bayes, LinearSVC, kNN, RandomForestClassifier, Perceptron, SGDClassifier and LogisticRegression. Analyzed the performance outcome of each of them. ExtraTreeClassifier seemed to be giving the best results.

2. Steps followed for model development:

- **Read the data set**
  Read the training data, labels and test data from the files train.dat, train.labels and test.dat using "loadtxt" from numpy.
- **Split into Training and Evaluation Data**
  Split the labeled data into train and test subsets, 80% for training and 20% for evaluation, in order to evaluate the predictive quality of the model, before making predictions on the unseen data set.
- **Standardization and Normalization**
  Standardization and Normalisation did not contribute to the F1 score improvement. Hence, commented out these steps.
- **Random state**
  Set the random state value for the test, train split set and classifier model so that we get the same result set and output each time the program is run for the training data.
- **Dimensionality Reduction**
  Use a pipeline to apply a list of transforms and a final estimator. SelectKBest module was used for dimensionality reduction and to select features according to k highest scores. Tried other dimensionality reduction techniques like the PCA, SVD and LDA. But got the best results with feature selection method.

- **Classification**
  Applied ExtraTreeClassifier for classification with the necessary parameters. It fits a number of randomized decision trees on various sub samples of data set.
- **Handle imbalanced data set**
  To handle the imbalanced data set, multiple options were used like the Over_Sampling, Under_Sampling, Random_Sampling and SMOTE options of the imbalanced-learn extension. However, none of them contributed to improving the F1-score significantly.
- **Performance Metrics**
  To check if the predictions are good for the training data set, calculated the F1 score, Recall and Precision after each run. As we had an imbalanced data set. It considers both the precision and the recall to evaluate the model. Trained the model with the split train data and then measured the f1 score using the test subset split from the train data.
- **Test Set Prediction**
  After getting a satisfactory F1 score with the train data, predicted the output for the test data and wrote the result to a file named format.dat. Redid the whole exercise multiple times to achieve the best results.

3. Methodology of choosing the approach and associated parameters

Tried with multiple classifiers and dimensionality reduction techniques and fine-tuned the ones that shows optimum performance.

Fine Tune Feature Selection parameters:

- **Score_func –** f_classif is the default score_func that works best for classification tasks. Chi2 score_func did not give as good results.
- **K –** For the number of top features to select, tried different numbers between the range 1 to 887. K value range between 50 to 70 gave the best results. Chose 60 as the parameter.

Fine Tune Classification parameters:

- **N_estimators –** This is the number of trees in the forest. Ran the code with multiple estimator values between the range of 150 to 400. 200 estimator gave a good score.
- **Min_samples_split –** This is the minimum number of samples required to split an internal node. Value 6 was chosen.
- **Random_state –** A non-zero random_state value was set to be able to reproduce the results in the after each run.
- **Max_features –** max_features value was set to “None”, indicates max_features = n_features.
- **Class_weight –** Class_weight was set to “balanced_subsample”. If not set, all classes are given weight 1. In this case, weights are computed based on the bootstrap sample for every tree. After setting this value I could see the rare classes of 10 and 11 in the final output result set.