**MS4610**

**Introduction to Data Analytics**

**Project Report: Loan Default Prediction**

**Team Members:**

Ranjith Kumar R M BS17B024

D V Sudharshanan ME17B137

Naveen Prashanna G ME17B156

Deva Krishna ME17B175

Ujjal Nandy ME17B085

**Data Preprocessing:**

From the correlation matrix of the given dataset, it is found that the features score5 and expense are perfectly correlated hence the feature score5 is removed from the dataset as it might lead to dummy variable trap.

For missing values in the dataset, the following methods are used,

* The records with missing labels are removed completely.
* An iterative imputer is used to fill in the missing values for the continuous features. Iterative imputation refers to a process where each feature is modeled as a function of the other features, e.g. a regression problem where missing values are predicted. Each feature is imputed sequentially, one after the other, allowing prior imputed values to be used as part of a model in predicting subsequent features. It is iterative because this process is repeated multiple times, allowing ever improved estimates of missing values to be calculated as missing values across all features are estimated.
* A KNN imputer is used to fill in the missing values for categorical features. In this case instead of an iterative approach a KNN model is used to model the missing features in terms of other features. Since categorical features are not continuous this method proves to be better than the iterative imputer approach. A k value of 5 is used for this model.

**Data Imbalance:**

In the given dataset the number of records with label 0 is 93%, this imbalance causes the model trained to be biased towards the label 0. In order to avoid that several techniques were experimented. The basic methods to solve this is either by Undersampling or Oversampling.

Undersampling involves randomly sampling from the majority dataset until the size of sampled data becomes equal to the number of records in the minority dataset. The base model gave poor performance upon applying this method, this can be attributed to the fact that we’re losing valuable data on undersampling. Hence, oversampling is preferred.

Oversampling can be done using two methods SMOTE(Synthetic Minority Over-sampling Technique) and Random Oversampling. In SMOTE, we interpolate the available points in the minority class to generate new points from this interpolation. In Random oversampling, we sample randomly from the minority class with replacement until the number of records for the minority class becomes equals to the number of records in the majority class. On experimentation, the Random oversampling technique gave better results than SMOTE. This can be due to the inability of interpolation to generalize the dataset properly.

**Choosing a classifier:**

The given dataset is split into training set and test set with a test set size of 0.2. Several classifiers such as KNN, XGBoost, Random Forest, SVC, are used to fit the dataset. Out of the used models Random Forest classifier gave better results both in terms of accuracy and f1 score.

In this case f1 score is chosen as a metric over accuracy due to the imbalance in the dataset. The Random Forest Classifier is further chosen for hyperparameter tuning.

**Hyperparameter Tuning for RandomForestClassifier:**

For a RandomForestClassifier, hyperparameters whose tuning will elicit significant impact on learning are (search space associated with each hyperparameter is also provided)

* Number of trees in random forest
  + n\_estimators = [int(x) for x in np.linspace(start = 100, stop = 500, num = 10)]
* Maximum number of levels in tree
  + max\_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
  + max\_depth.append(None)
* Minimum number of samples required to split a node
  + min\_samples\_split = [2, 5, 10]
* Minimum number of samples required at each leaf node
  + min\_samples\_leaf = [1, 2, 4]

A grid with the above details regarding search space of hyperparameters under consideration is created. RandomizedSearchCV is deployed to perform a random search on the provided search space rather than probing all possible combinations of hyperparameters. The most important arguments in RandomizedSearchCV are n\_iter, which controls the number of different combinations to try, and cv which is the number of folds to use for cross validation (we use 10 and 5 respectively). More iterations will cover a wider search space and more cv folds reduces the chances of overfitting, but raising each will increase the run time. Machine learning is a field of trade-offs, and performance vs time is one of the most fundamental.

This yielded best hyperparameter set:

n\_estimators : 366

min\_samples\_split: 5

min\_samples\_split: 5

min\_samples\_leaf: 1

max\_depth: 60

bootstrap: true

The corresponding f1 score on test set is 0.8731