ANJISHNU MUKHERJEE
G01373990
amukher6@gmu.edu
dataminer username - noobmaster

Homework Assignment 2

Data Mining (CS 584), Fall 2022

2022-10-12

1 Abstract

The assignment consists of 4 distinct parts :

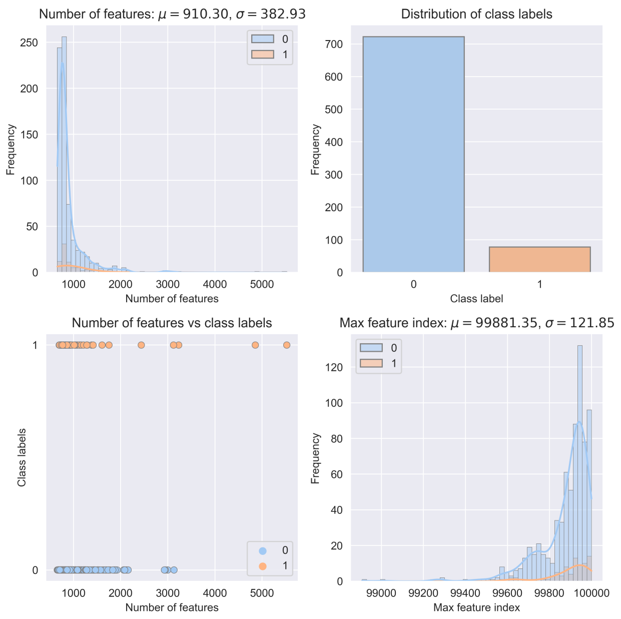
- Loading the data in sparse matrix representation.
- Dimensionality reduction
- Oversampling technique
- Cross Validation to find the best model

Once all the above steps are complete, we need to use the best set of parameters as obtained from the training set, to predict class labels on our test set. The resulting predictions file is uploaded to the dataminer website.

2 Methods

2.1 Input

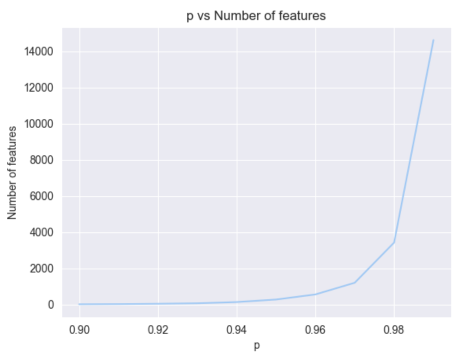
There are 800 training data points, each having a different number of feature indices where the value is 1. The number of features (with value 1) vary from 657 to 5524. The dataset is not balanced, less than 1% of the data points have the class label 1. The max value of feature index is 100,000 (i.e in any given row in the input, the maximum value is 100,000) and the distribution of the indices can be seen in Figure 1.



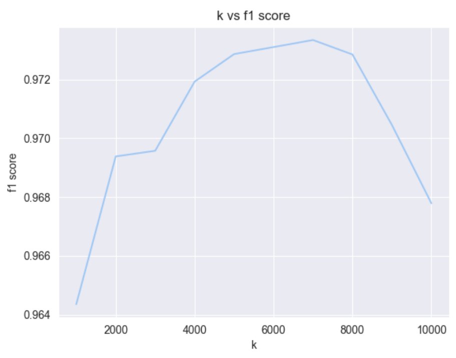
Since the maximum index goes upto 100,000 and there are 800 samples in the data, I can store all of the input in a matrix of size (800, 100000). But this is not very efficient in terms of memory or speed. So, I adopt a sparse matrix representation that is compatible with scikit learn functionality. I convert the input representation into the Compressed Sparse Row matrix format.

2.2 Preprocessing

But it does not make sense to use all the 100,000 features. Some of the features are all 0's indicating no-variance. I apply variance thresholding modelled using a Bernoulli distribution, because boolean features are binary random variables such that $Var[X] = p * (1 - p)$ where X is a feature. This reduces my data to about 14,000 features when I set p to 100,000 in the above equation.



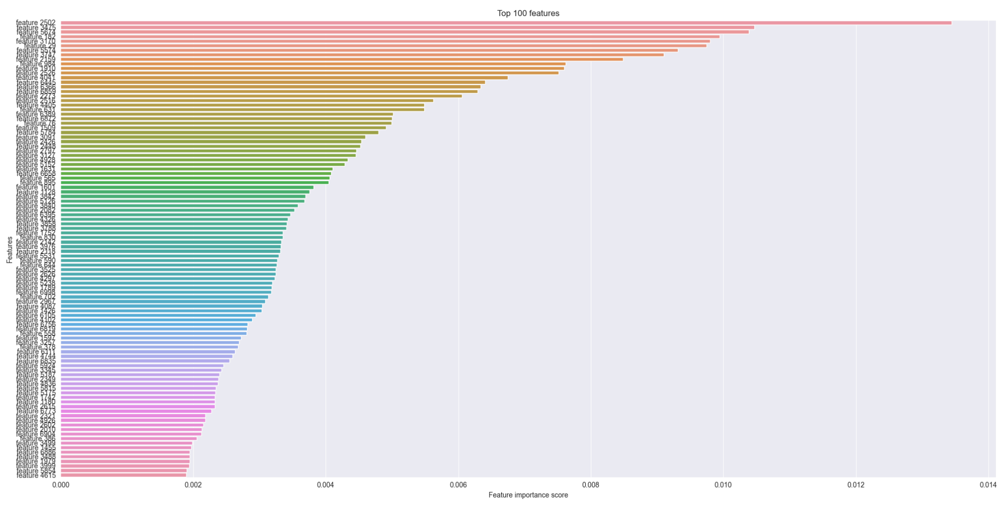
(a) p for vs number of features



(b) k for select-k-best feature selection vs f1 from naive bayes

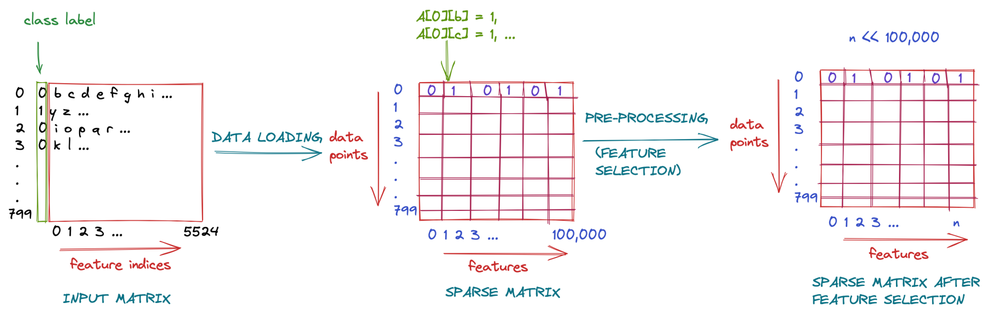
After this, I select the top 7000 features based on chi2 scoring function. I chose the value 7000 becaue after fitting my optimal NB classifier on this data, 7000 was giving the highest f1 scores.

Finally, I try one more feature selection procedure. I use a Random Forest Classifier as a feature selector. This allows me to reduce my feature set down even further. Depending on the other hyperparameters that I choose, I can have the final number of features as low as 250, or slightly higher (1000, 3000 or 5000). This process also gives some intuition about which feature might be important, acting as sort of an "explanation" method.



With this final set of features being a very small fraction of the original number of features (100,000), I would move into imbalanced class sampling next.

The process followed till now is illustrated below.



2.3 Imbalanced class sampling

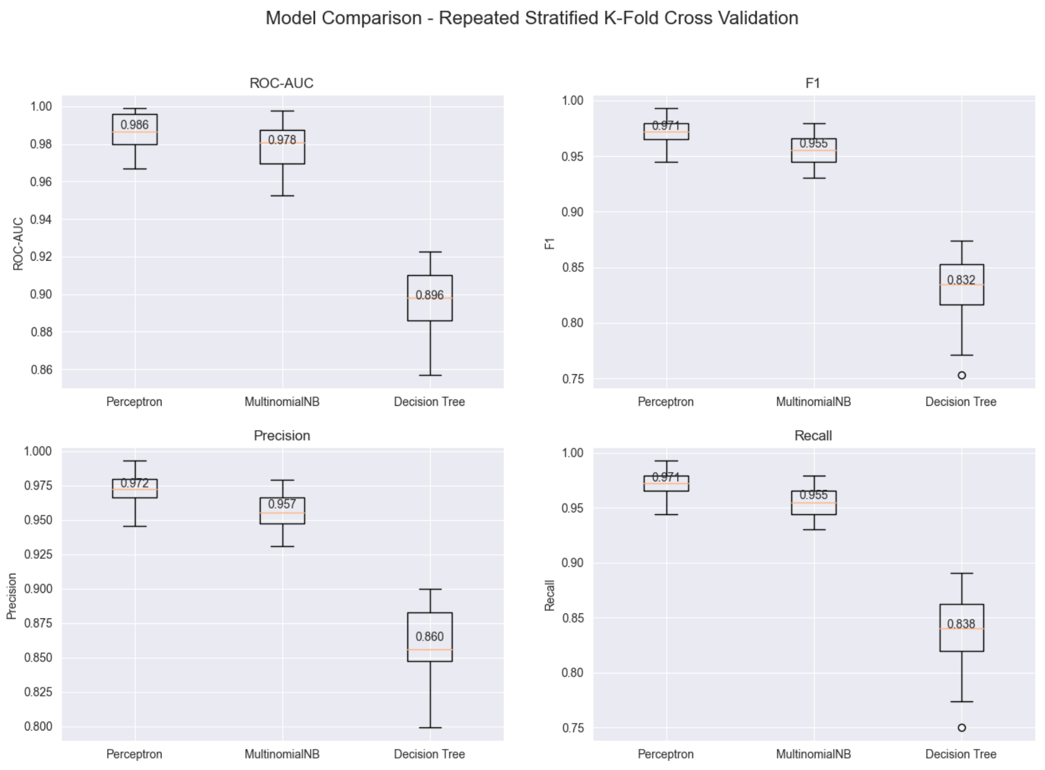
I used SMOTE Tomek which is a combination of over-sampling method (SMOTE) and undersampling method (removing Tomek Links). SMOTE stands for Synthetic Minority Over-sampling Technique and is a very well-known oversampling method but it has some common pitfalls, for example it can generate noisy samples by interpolating new points between marginal outliers and inliers. This issue can be solved by cleaning the space resulting from over-sampling. To do this, we perform under-sampling by removing Tomek Links.

Balanced Class distribution after SMOTETomek : (0: 722, 1: 722)

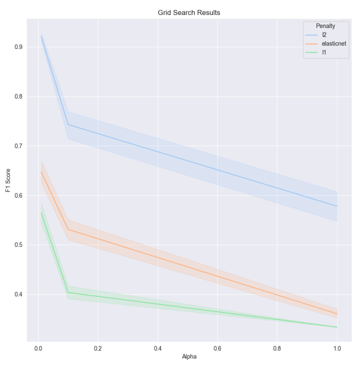
2.4 Grid Search

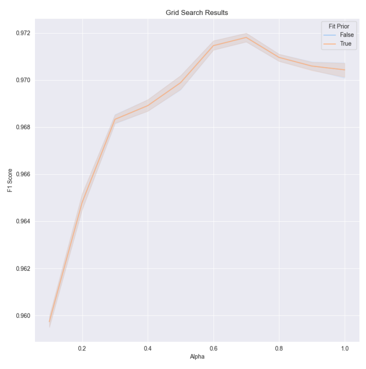
We are allowed to use three sklearn classifiers in this assignment - Multinomial Naive Bayes, Perceptron, Decision Tree.

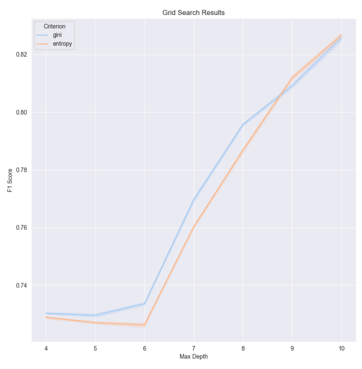
I first compare the three models on a non-optimal set of hyperparameters to understand which model might perform best in general without any fine-tuning. This study also gave me an idea about how the models perform for different metrics - Precision, Recall, F1, ROC-AUC (area under ROC curve).

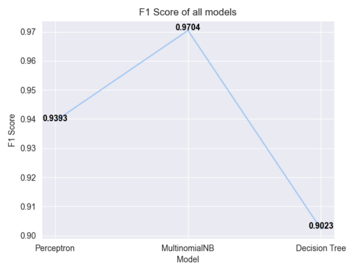


I grid searched over possible parameters for each of the classifiers individually first, before running cross validation experiments. The graphs in the following figure represent the results of this grid search process in the order Perceptron, Multinomial Naive Bayes and Decision Tree in that order respectively.

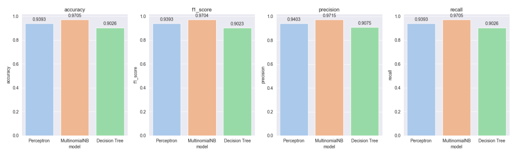








(a) cross validation f1 scores



(b) model comparisons

2.5 Cross Validation

Using the 3 best sets of parameters obtained from Grid Search above, I trained the three models using Repeated Stratified K Fold cross validation. Stratified cross validation method is used to account for the class imbalance in the dataset.

This process would give me the "best" model among these three. For me, for most experiments I ran, this was the Multinomial Naive Bayes. An example run producing my highest score on dataminer is included in the image.

2.6 Comparing with majority voting

To gain some more confidence in the predictions of my "best" model, I created a majority voting scheme among all three models such that if all the three models agree with the best model, only then test point gets class label as 1.

Using this, I get 23 test points with the class label as 1 out of the 350 provided points. This corresponds to my highest score of 0.67 on dataminer.
Irrespective of whatever else I tried within the constraints of the assignment, the score did not improve further on the test set.

3 Results

model	accuracy	f1 score	precision	recall
Multinomial NB	0.970450	0.970432	0.971479	0.970450
Perceptron	0.939294	0.939255	0.940295	0.939285
Decision Tree	0.902276	0.902276	0.907472	0.902605

Timing measurements :
Configs loaded in 0.0043 seconds
Datasets (train and test) loaded in 1.0053 seconds
Imbalance class sampling completed in 0.6455 seconds
Dimensionality reduction completed in 19.8089 seconds
Cross validation loop completed in 5.8019 seconds
Test set prediction generation completed in 0.1463 seconds

4 Conclusion

After running all these experiments, I got 0.97 highest score on validation set and 0.67 highest score on test data. A set of the best params is included in the config file submitted. Instructions to run are included in the README file in src.