EE 769 : Introduction to Machine Learning
Guide : Prof. Amit Sethi

Simple Classification

Based on the given attributes predict the selling status of a house

Name : Debanjan Das

Roll : 173050069

Table of Contents

1. Introduction

In this assignment, we have to perform three class classification. Based on the given attributes about houses in a particular country, we have to predict the class in the last column, which is "SaleStatus". For this purpose, we are given training data in the file trainSold.csv, and testing data in the file testSold.csv. The latter has the last column missing, for which we have to generate labels.

Tools used : Python3, Jupyter Notebook, Scikit-learn, Matplotlib, Numpy, Pandas.

2. Strategy for Data Pre-processing

Data preprocessing is the technique of handling raw data and transform it to a machine understandable format, e.g, some features in the given data had text values, but machine understands number better. So, we convert text values to numerical values.

During preprocessing phase I faced the following issues:

- Presence of NaN values in some features and
- Some feature has text values.

I handled the presence of NaN values by replacing them with the **mode or mean** (depending on the datatype of the feature) of the feature values. The following code snippet has achieved the goal.

```
for e in error_features:
        q = dcn[e]     #dcn is dictionary containing all the training data
        dcn[e] = [str(x) for x in q]
        for i in range(len(q)):
                if dcn[e][i]=='nan':
                        dcn[e][i] = dcn_stat[e]
#dcn_stat is a dictionary containing null valued feature names and corresponding
mean or mode value
```

Text valued features are handled by performing **encoding** on them. There are two types of encoding I have used in this assignment, they are : **a. Label Encoding** and **b. One Hot Encoding**.

First, I tried Label Encoding. But, one issue with this encoding is that machine learning algorithms will assume that two nearby values are more similar than two distant values. Obviously this is not the case. Thus my next option was to try out One Hot Encoding.

In One Hot Encoding only one attribute of a feature will be equal to one (hot) and others will be zero (cold). In this encoding, the bias due to a false notion of ordinal relationship has been completely eradicated.

Code snippet for One Hot Encoding is given as follows :

```
categorical_attr = housing.select_dtypes(include=['object'])
housing_t = housing
for i in categorical_attr.columns:
        x = pd.get_dummies(housing_t[i])
                housing_t = pd.concat([housing_t,x],axis=1)
         del housing_t[i]
```

While pre-processing test data I observed that the number of features got reduced than that of train data set. So, I added the missing features in the test dataset.

After preprocessing I splitted the dataset in a certain ratio (80% train & 20% test) as train data and test data for performing validation. The splitting has been done randomly on the dataset for circumventing the data snooping bias.

3. Classifiers

I have used four classifiers : a) Support Vector Machine(SVM), b) AdaBoosting Classifier, c) Random Forest Classifier and d) Gradient Boosting Classifier. Out of these Gradient Boosting Classifier worked best on the validation data achieving **97.602%** accuracy.

a) Support Vector Machine :

It performed well on the validation data with an average accuracy of **90.4692308%**. The possible reasons behind this performance are stated as follows :

- The dataset was not highly skewed or imbalanced. That is instances of one class did not outnumbered the no. of instances of other classes.
- Number of features after one hot encoding is not huge compared to the number of data instances.
- I scaled and normalized the features before prediction.
- The number of data points were less.

b) AdaBoosting Classifier :

AdaBoost classifier achieved an average accuracy of **95.89041%**. The possible reasons behind this performance are stated as follows :

- The combination of all the weak learners, in this case it is decision trees.
- By iterating the weights on the weak learners it gets better and better over time. It assigns the negative weights to the wrong predictors and assigns higher weight to the right predictor.

c) Random Forest Classifier :

Random Forest classifier clinched at the average accuracy of **96.3491%.** The possible reasons behind this performance are stated as follows :

- As the assignment was a multi-class problem thus it is suited for random forest classifier.
- Random Forest works well with a mixture of numerical and categorical features which is the case in our given dataset.
- As our dataset might have been created from some surveying thus it increases the probability of outliers. In such cases random forest works well.

d) Gradient Boosting Classifier :

The Gradient Boosting classifier secured an average accuracy of **97.602739%**. The possible reason for this better performance on the given dataset are :

- Gradient boosting is built on weak classifiers, i.e, high bias and low variance, somewhat similar to the random forest.
- It is a additive model, and it reduces the biases by aggregating the results from each weak classifiers.
- As it is a multi class problem thus gradient boosting performs well on the dataset.

Gradient boosting classifier has *not been taught in the class* and it has achieved an accuracy of 97.602% (*>95%*).

4. Hyperparameters

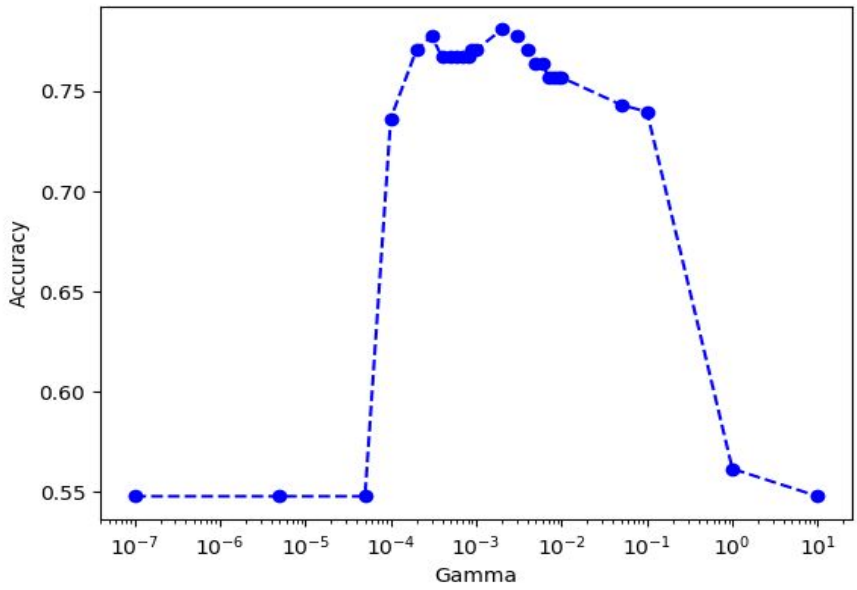
The hyperparameter of a model is the external configuration of the model; its value can not be estimated from a given dataset. I tuned the hyperparameters of each of the classifiers I used in the assignment.

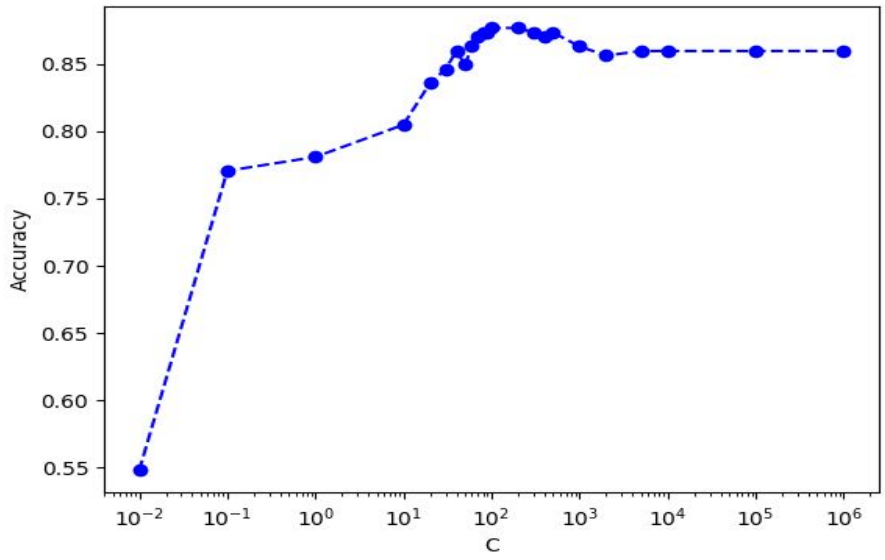
Instead of using the grid search I extrapolated a set of values that I observed is giving best results for the corresponding classifier. The reason for executing this method is to reduce the computation time. Because, to completely visualize the validation curve we need to plot values within a sufficiently large interval where incrementation amount should not be very large. But, this would result in plotting of many unnecessary points which will in turn increase the computation time. The hyperparameter tuning for the classifiers are given as follows :

a) Support Vector Machine :

In SVM I tuned gamma and C, C and gamma are parameters of non-linear SVM with Gaussian radial basis function kernel. C is the parameter for the soft margin cost function, which controls the influence of each individual

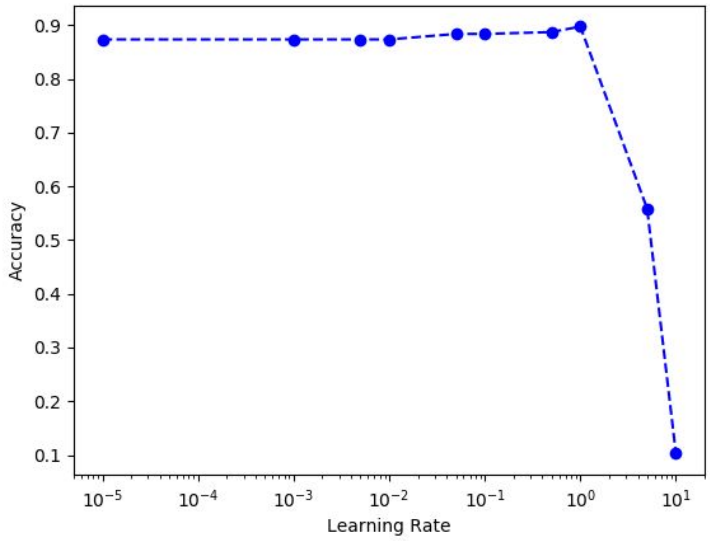
support vector; this process involves trading error penalty for stability. Gamma is the free parameter of the Gaussian radial basis function. The validation curves are given as follows :

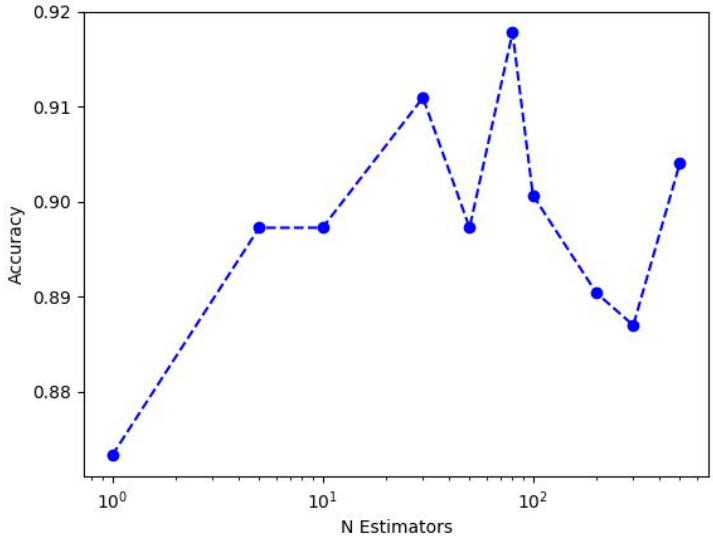




b) AdaBoosting Classifier :

In AdaBoosting classifier I tuned hyper- parameters are the learning rate and n_estimators. Learning rate reduces the contribution of each classifier by the amount of learning rate. The number of weak learners are controlled by n_estimators. I used decision tree classifier as weak learners. The validation curve is given as follows :

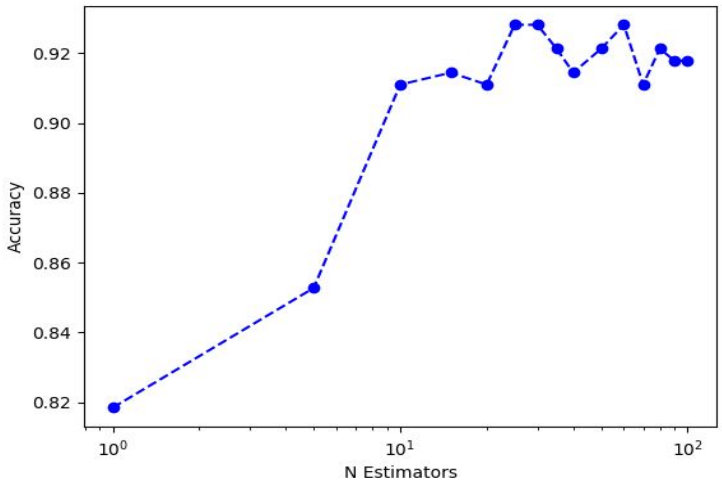


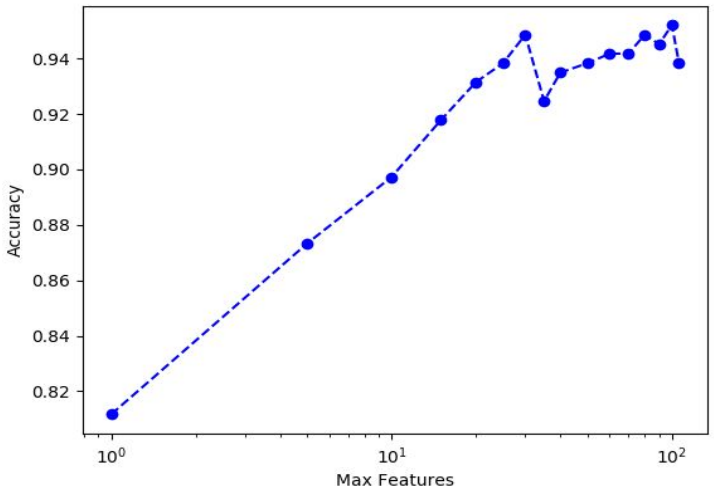


c) Random Forest Classifier :

Generally random forests are tuning free but still to visualize the validation curve I tuned the following hyperparameters :

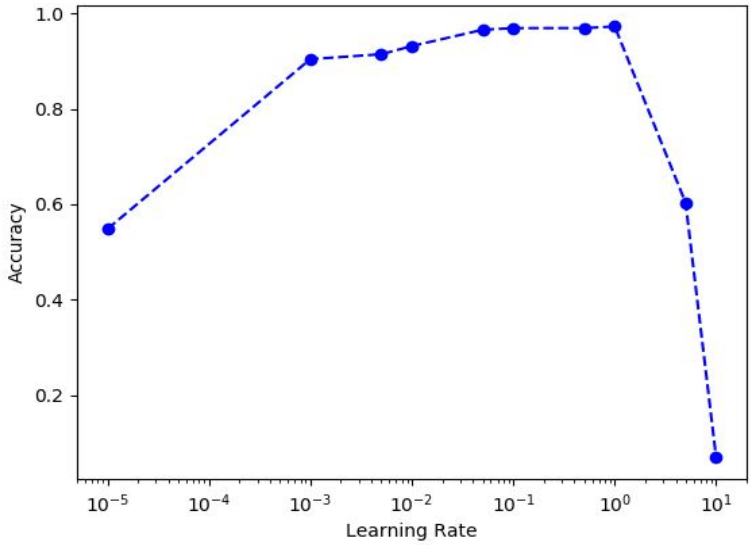
- N_estimators : Number of trees in the forest.
- Max_features : The number of features to look at while searching for the best split. The validation curves are given as follows :

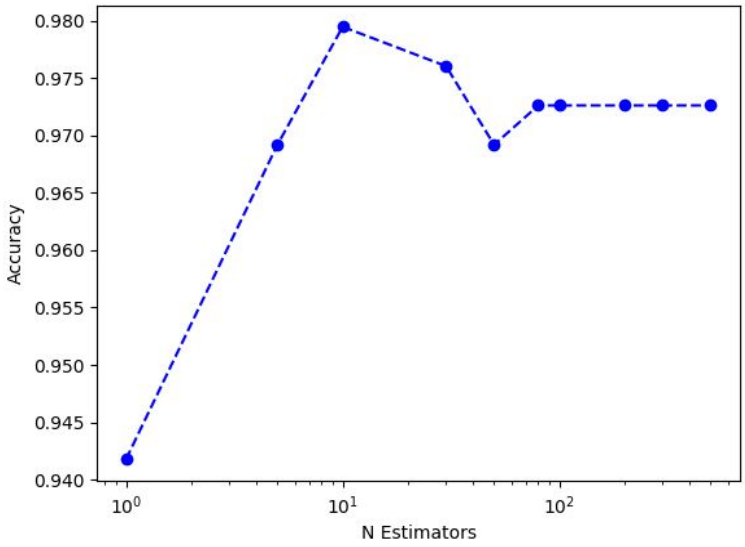




d) Gradient Boosting Classifier :

Learning rate and n_estimators are the hyper-parameters that has been tuned. Learning rate reduces the contribution of each tree by the amount of learning rate. N_estimators is the number of boosting stages to perform. The validation curves are given as follows :





5. Conclusion

Out of the four classifiers Gradient Boosting classifier achieved best accuracy. Thus it has been used as the final model for the prediction on test data. The trained model is saved as *finalModel1.pkl* in the same directory as that of *train.py.*

The final model achieved **97.602%** accuracy on the validation data.

6. References

[1] www.scikit-learn.org

[2] www.en.wikipedia.org

[3] www.stats.stackexchange.com