EE 769 - Introduction to Machine Learning

Dhananjay kr. Sharma
173050046

Report : Assignment 1: Simple Classification

Tools Used:

1. Pandas
2. Numpy
3. Scikit learn(sklearn)
4. Pickel
5. Csv
6. Matplot

1) What strategy worked best for data pre-processing and why? Mention any strategies tried that didn't work.

Preprocessing on training dataset:

First of all I read the input train.csv file into 'train' dataframe using **pandas** library function. Then I stored its '**SaleStatus**' column into a **'label'** data frame and then deleted 'SaleStatus' column from the training data. After this I filled all the empty(nan) fields with median value of that column data. Now I encoded label dataframe using **LevelEncoder** because data-type of label data-frame is of string type. Since, some features in train data-set are of **object data-type** so I performed **OneHotEncoding** on train data. After this I scaled down every column data to reduce processing time using following formula:

train[column] = (train[column] - train[column].mean()) / (train[column].max() - train[column].min())

Now I divided the input train data frame and label into 80:20 ratio using train_test_split function present in sklearn.model_selection and stored splitted data into four variable like below:

```
X_train, X_test, Y_train, Y_test = train_test_split(train,labels, test_size=0.2)
```

Preprocessing on test-dataset:

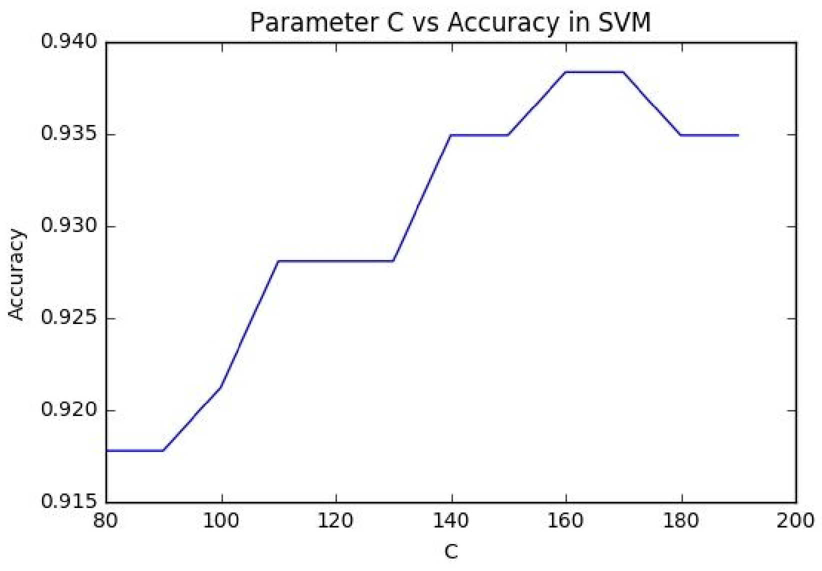
After generating different model.pkl file from train.py, I am predicting results for the test data set using model.pkl file. But to predict, we need to have similar data structure of test data set like train dataset. So first of all, I am reading test.csv file in test data frame and then filling empty filled with median value of the respective column. After this , I performed **OneHotEncoder** on test data just like train data set. After encoding, I am adding columns in test data frame if a column is not there in test data set. And finally, I scaled test data set just like train data set.
Now to calculate accuracy of test data, we need to have correct labels for test data that we have in gt.csv file. I read gt.csv file in label data frame and extracted ‘SaleStatus’ from that and calculated accuracy using predicted label and the actual label from gt.csv.

2) Which classifier worked the best? Why do you think those were appropriate for this data? What else was tried that didn't work?

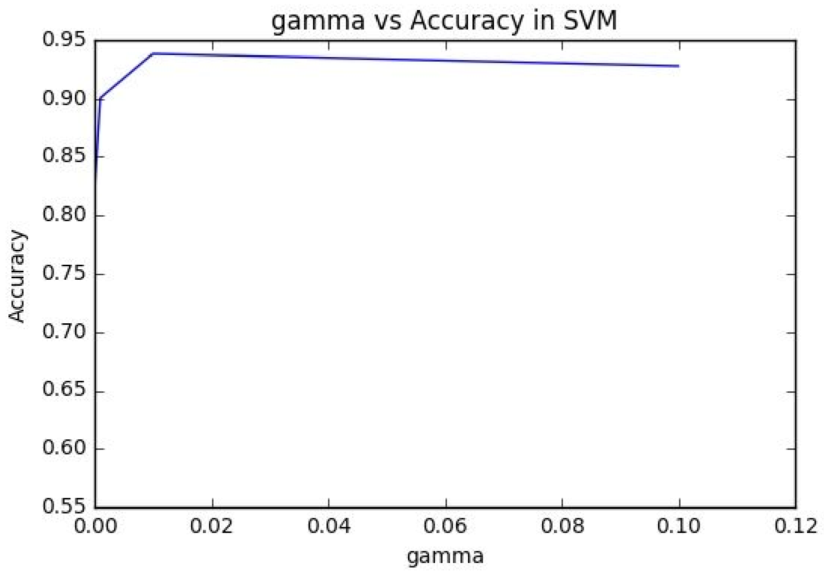
I tried four classification model on training data set : SVM model, Gradient-Boosting model, Decision-tree model and Random forest; but among all above classifier, **Gradient boosting model has highest accuracy with 97.26** with parameter value **learning rate=0.15** and **n_estimator=160.** SVM gives good accuracy when number of training sample is large.

3) Which hyper-parameters were tuned and how? You may include hyper-parameter tuning graphs here.

1. **SVM Classification:** Support Vector Model works well when we have large set of training data set but here we have only 1500 of samples so its performance is not that good. SVM has 3 important regularization parameter: i.) gamma, ii.) C, iii.) Kernel. For the given input training data set, after performing on many different values of parameter, I got following result:
   Tuning parameter C in SVM:



Tuning parameter C in SVM:



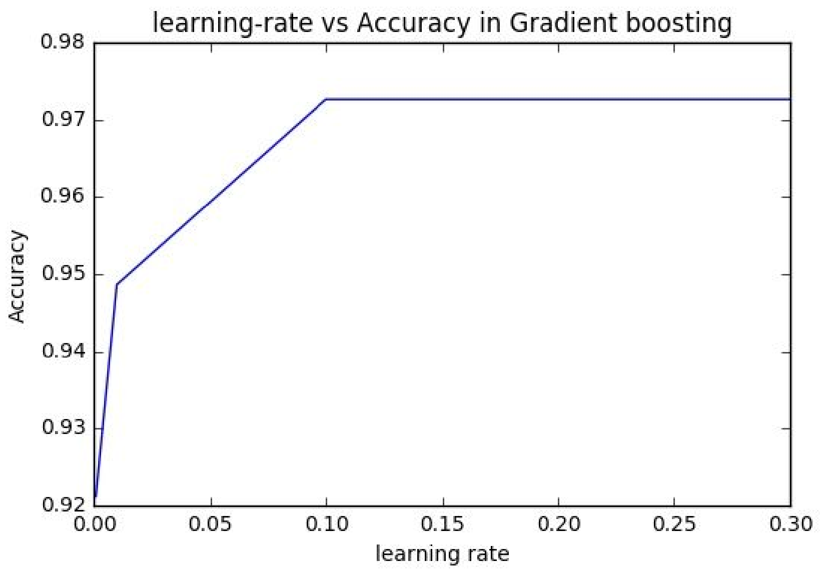
In **SVM**, I got **maximum accuracy of 94.86%** on training set with parameter value: **gamma=0.01** and **C=160.**

2. **Gradient Boosting Classification Model:** Gradient boosting model build trees one at a time, where each new tree helps to correct errors made by previously trained tree. With each tree added, the model becomes even more expressive. There are typically two parameters - depth of trees and learning rate. Following is the code for Gradient boosting classification:

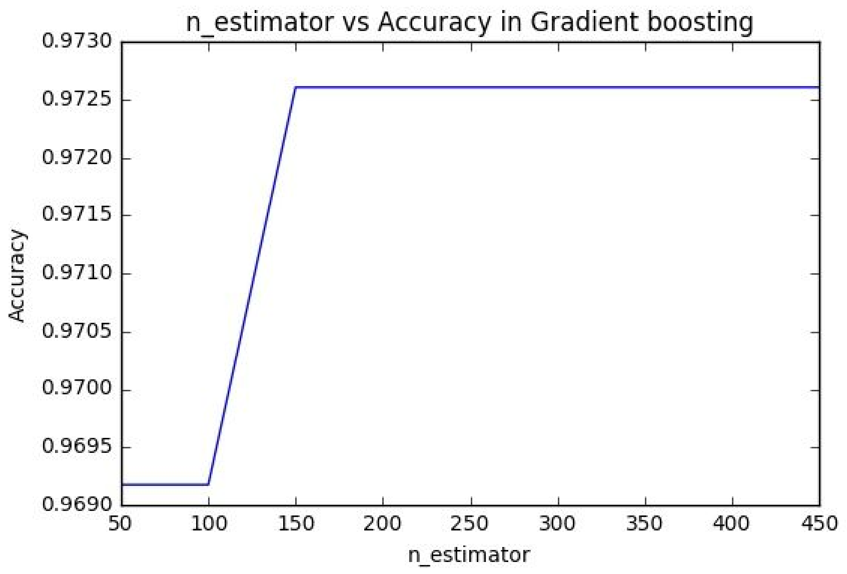
```
parameter={'n_estimators':160,'max_depth': 3, 'subsample': 1.0,
'learning_rate': 0.15,'min_samples_leaf': 1, 'random_state': 3}

    clf_g = ensemble.GradientBoostingClassifier(**parameter)
    clf_g.fit(X_train,Y_train)
    pred_grd= clf_g.predict(X_test)
```

Tuning Parameter learning rate in Gradient Boosting Model



Tuning Parameter n_estimator in Gradient Boosting Model:

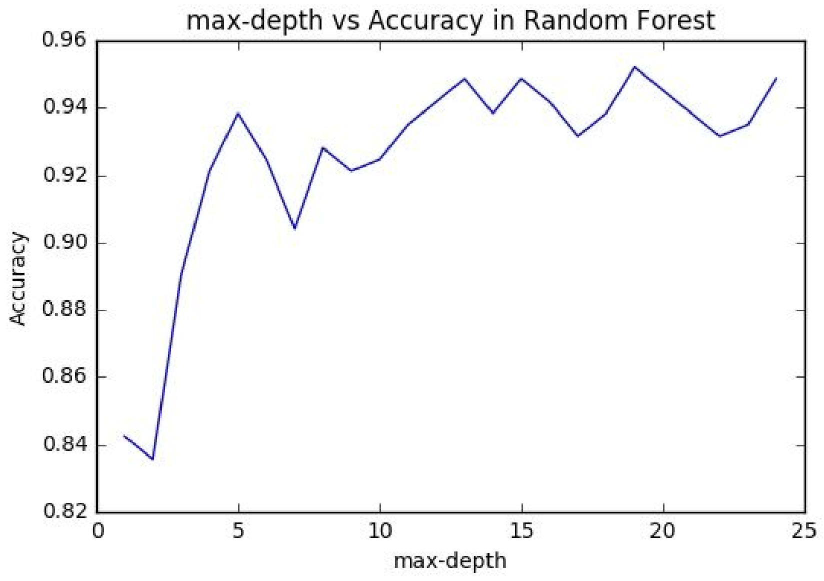


In **Gradient boosting** , I got **maximum accuracy of 97.26%** on training set with parameter value: **learning rate=0.15** and **n_estimator=160.**

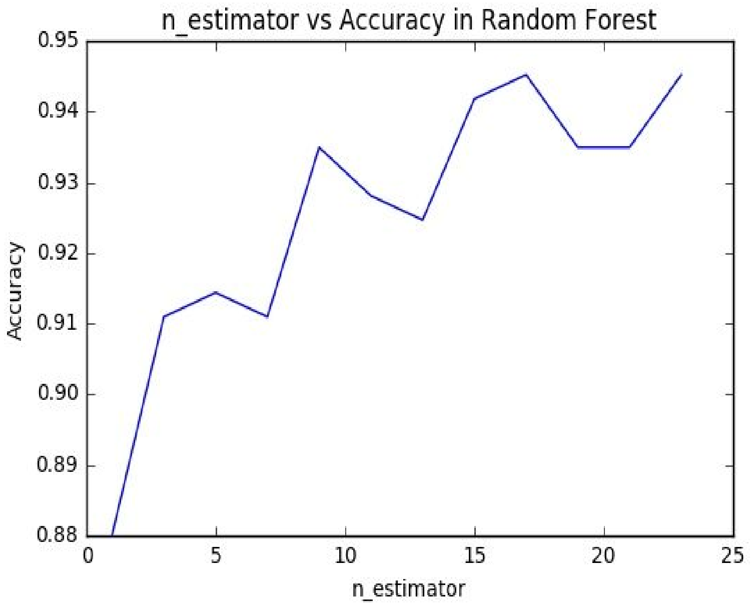
3. Random Forest Classification model:

There are typically two parameters in RF - number of trees in the forest and maximum depth of the tree.

Tuning Parameter max_depth in Random Forest Model:



Tuning Parameter n_estimator in Random Forest Model:



In **Random Forest model**, I got **maximum accuracy of 93.15%** on training set with parameter value: **max_depth=14** and **n_estimator=14.**

Source used:

1.http://scikit-learn.org
2. https://stackoverflow.com/
3.https://machinelearningmastery.com/