Machine Learning Engineer Nanodegree LONGCHANG CUI

Jan 28th, 2017

Alzheimer Analysis with Supervised Learning

Definition

Project Overview

Alzheimer's disease is a progressive brain disorder that damages and eventually destroys brain cells, leading to memory loss and changes in thinking and other brain functions. It is the most common type of dementia, and Alzheimer's disease accounts for 60 to 80 percent of dementia cases. Those with Alzheimer's live an average of eight years after their symptoms become noticeable to others, but survival can range from 4 to 20 years, depending on age and other health conditions.Alzheimer's has no current cure. Thus, it is critical to prevent the disease before it occurs.

It is commonly known that aging is the primary factor that results in Alzheimer disease. However, aging is not the only factor that can cause the disease. There are other factors that we can examine to help developing a better understanding of Alzheimer disease. In this project, I will explore the OASIS[1] dataset and use various supervised learning algorithms to discover the different perspectives on the disease.

The data was obtained from Open Access Series of Imaging Studies[2]. It includes the values of intracranial volume, brain volume, Mini-Mental State Examination result, Clinical Dementia Rating and the demographics. It consists of a cross-sectional collection of 416 subjects aged 18 to 96. All the subjects are right-handed and include both men and women. 100 of the included subjects over the age of 60 have been clinically diagnosed with very mild to moderate Alzheimer's disease (AD). Additionally, a reliability data set is included containing 20 non-demented subjects imaged on a subsequent visit within 90 days of their initial session.

The data set includes demographics features, such as are gender(M/F), handedness(hand), age, education(Educ) and socioeconomic status(SES). Education values correspond to the following levels of education: 1: less than high school grad., 2: high school grad., 3: some college, 4: college grad., 5: beyond college.

The data set also includes estimated total intracranial volume (eTIV) (mm3), Atlas scaling factor (ASF), Normalized whole brain volume (nWBV). The clinical features are Mini-Mental State Examination (MMSE), Clinical Dementia Rating (CDR; 0 = non-demented; 0.5 = very mild dementia; 1 = mild dementia; 2 = moderate dementia). All participants with dementia (CDR > 0) were diagnosed with probable AD. It is the target feature that will be used in the predictions. It is also the most important feature of the data since the CDR can be used to determine the classification results.

---

(1) The Open Access Series of Imaging Studies (OASIS) is a project aimed at making MRI data sets of the brain freely available to the scientific community. By compiling and freely distributing MRI data sets, we hope to facilitate future discoveries in basic and clinical neuroscience.

(2) Open Access Series of Imaging Studies Data http://www.oasis-brains.org/app/template/Index.vm

Problem Statement

The goal of the project is to observe how different features could potentially cause dementia and use the features of data to train the supervised learning model to predict the Alzheimer disease.

The values of the Clinical Dementia Rating in the dataset determine whether the subjects have Alzheimer disease or not. The CDR feature has the value of 0.5, 1.0, 2.0. The value 0 means non-demented; 0.5 means very mild dementia; 1 means mild dementia; 2 means moderate dementia. In other words, if a subject who has the CDR value greater than zero, then the subject has dementia. Since the level of dementia is not the major interest of the project, we need to convert the CDR values into binary values which are 0 and 1. The value 0 in CDR means the subject is not demented; the value 1 in CDR means the subject is demented.

The machine learning algorithm Gaussian Naive Bayes(GaussianNB), Decision Tree Regression(DT), and Supported Vector Machine(SVM) will be implemented based on the CDR values. These algorithms can be used to provide the predictions of diagnosing dementia. I will choose the machine learning algorithm that gives me the higher F1 scores. However, to avoid overfitting, I need to change different training size and set different conditions on several features to measure each algorithm's performance. The overfitting algorithm will not be used to generalize the result of the prediction.

Metrics

The F1 score is useful in this project because the prediction of the Alzheimer Disease is binary classification.

$$F1\ Score = 2 \cdot \frac{precision * recall}{(precision + recall)}$$

$$Recall = \frac{(True\ Positive)}{(True\ Positive + False\ Negative)}$$

$$Precision = \frac{(True\ Positive)}{(True\ Positive + False\ Positive)}$$

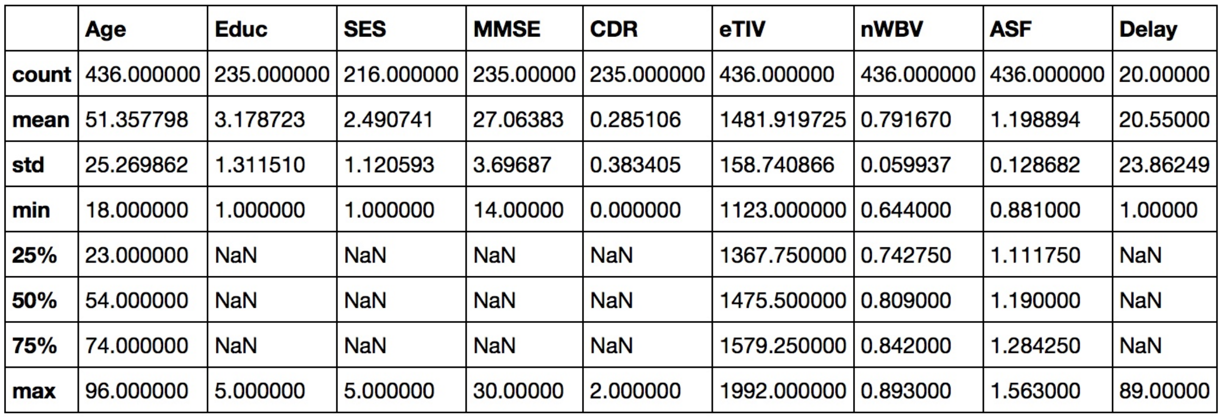
The precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. Both precision and recall are based on the measure of relevance.

In this project, the True Positive is the machine learning algorithm that correctly predict s the number of subjects who have dementia. The False Positive is the algorithm that incorrectly predicts the number of subjects who do not have dementia but classified as dementia. The False Negative is the algorithm that predicts the number of subjects who do have dementia but classified as not having dementia.

The F1 score is a good measurement for observing the accuracy of the supervised machine learning algorithm.

Analysis

Data Exploration



	Age	Educ	SES	MMSE	CDR	eTIV	nWBV	ASF	Delay
count	436.000000	235.000000	216.000000	235.00000	235.000000	436.000000	436.000000	436.000000	20.00000
mean	51.357798	3.178723	2.490741	27.06383	0.285106	1481.919725	0.791670	1.198894	20.55000
std	25.269862	1.311510	1.120593	3.69687	0.383405	158.740866	0.059937	0.128682	23.86249
min	18.000000	1.000000	1.000000	14.00000	0.000000	1123.000000	0.644000	0.881000	1.00000
25%	23.000000	NaN	NaN	NaN	NaN	1367.750000	0.742750	1.111750	NaN
50%	54.000000	NaN	NaN	NaN	NaN	1475.500000	0.809000	1.190000	NaN
75%	74.000000	NaN	NaN	NaN	NaN	1579.250000	0.842000	1.284250	NaN
max	96.000000	5.000000	5.000000	30.00000	2.000000	1992.000000	0.893000	1.563000	89.00000

Figure 1. Statistical description of the OASIS data set

The data set includes age, education, socioeconomic status (SES), estimated total intracranial volume (eTIV) (mm3), Atlas scaling factor (ASF), Normalized whole brain volume (nWBV). The clinical features Mini-Mental State Examination (MMSE) and Clinical Dementia Rating(CDR).

From the Figure 1 we can conclude that the average age of the subjects are 51 years old, the estimated total intracranial volume is 1481, Education level is 3.17, Clinical Dementia Rating is 0.29 and the Normalized whole brain volume is 0.79. Some of the columns does have the “NaN” values in median and percentiles. The “NaN” value denotes the empty value in the data. In other words, the information of the features is missing. We can also see that the total number of the subjects is 436, but from the ‘count’ row we can notice that several columns do not match the number. It is also due to the missing values in the features. Since we can not simply delete the subjects who have empty values, it is appropriate to solve this problem by filling in the mean values in the missing table.

The delay values are useless in our prediction since it only has 20 values. It cannot be used to generalize the prediction result. Thus, we should drop the Delay column.

Furthermore, the gender(M/F) column is missing here. It is because the values of the gender are not numerical values. In the following procedure, I will implement a algorithm and convert those values to numerical values so that we can use the gender feature in the machine learning algorithms.

Exploratory Visualization

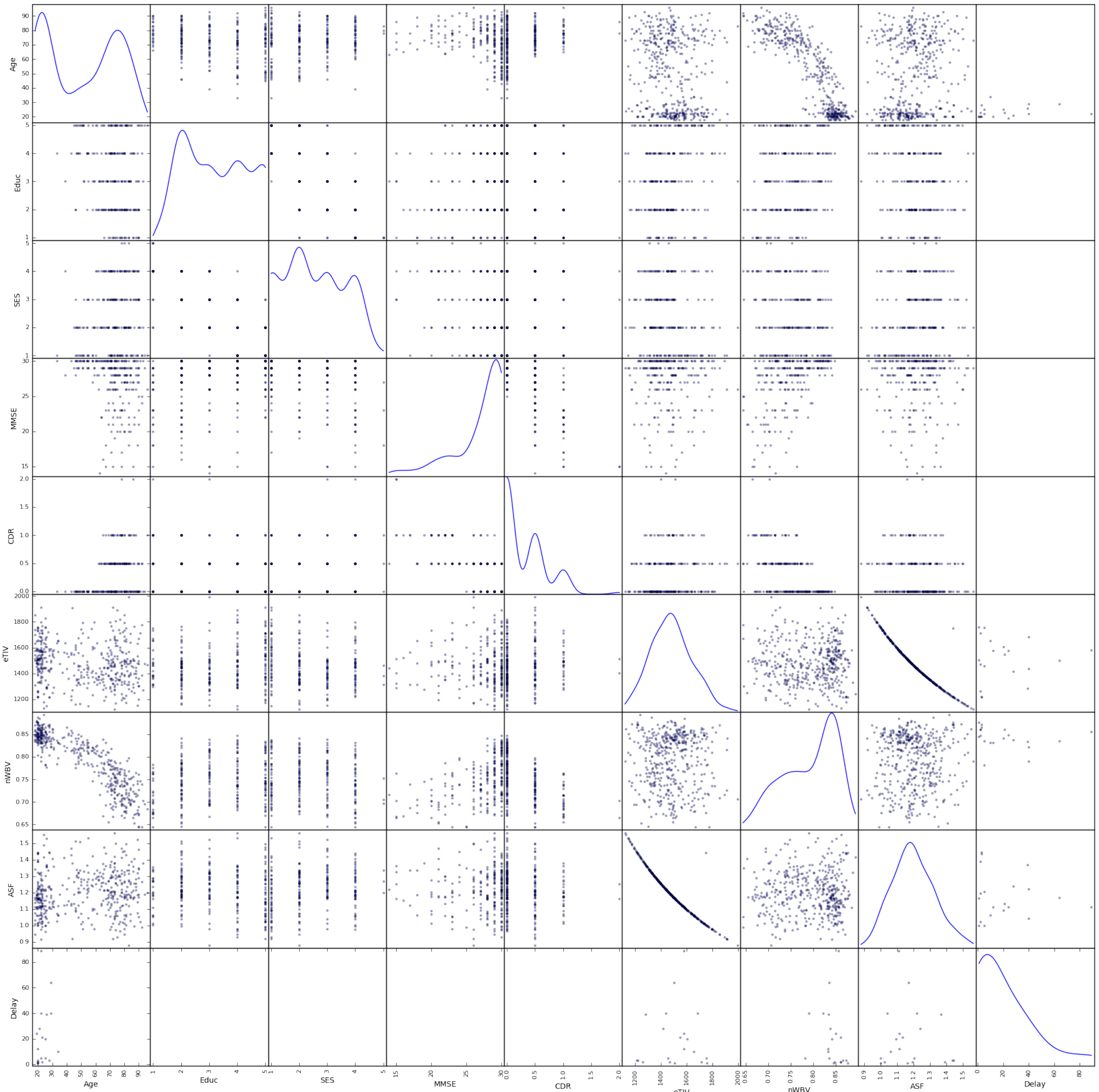


Figure 2. Scatter matrix of each feature.

The Figure 2 is a scatter matrix graph of the relationship of the each feature. From the graph above, we can notice that eTIV and ASF have certain degree of correlation which is not surprising since the value of total intracranial volume is estimated by the atlas scaling factor. However, the Age and the nWBV values are also showing some degree of correlation. It clearly shows that when the age increases, the normalized whole brain volume(nWBV) decreases accordingly.

We can also observe that the Delay feature almost has no relationship with most of the features. The OASIS website did not mention anything about the Delay feature and it does not make any sense to train the data with the feature Delay. Thus, we can also make the same conclusion as we made in the section of the Data Exploration. The Delay feature should be dropped.

Algorithms and Techniques

The machine learning algorithms that will be used in the project are Gaussian Naive bays(GNB), Decision Trees(DT), and Support Vector Machines(SVM). Since predicting the Alzheimer Disease is a binary classification task, these algorithms are efficient to use.

The Gaussian Naive Bays has light memory usage and only requires a small amount of training data to estimate the prediction. In other words, it performs well on the predicting even if we have very small training size. However, this algorithm assumes the features are independent of each other. Thus, the algorithm will perform poorly if the assumption is violated. As we can see from the earlier scatter matrix graph, there are not too many correlations between each feature hence the GNB would perform well in our data set. Since the GNB does not have many parameters, the default parameters will be used in the project.

The Decision Trees would also make a great prediction in our data set. First of all, the tree has a logarithmic running time complexity, and it performs well specifically in binary classification. The disadvantage of this algorithm is that it needs to be rebuilt if there is a variation in the original data and sometimes it will create biased trees if some features dominate in the data set. The DT parameters that will be modified are min_samples_split and max_depth. The max_depth of the tree can be modified to test the accuracy of the predictions and min_samples_split is useful when we train the data.

Although the Support Vector Machines grows exponentially in training time, it can tolerate errors and performs well even if there is noise in the data set. If we want to avoid over-fitting, SVM would be the best choice for our data set. The SVM algorithm has many parameters that can be modified to implement the different model. The kernel, C, and gamma parameters will be used to predict the Alzheimer Disease. Kernel parameter has ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ methods. In this project, we can either use the sigmoid or rbf to test our data set since the features of input data do not have much linear correlations. The C parameter tells the SVM optimization how much it wants to avoid misclassifying in each training. The Gamma is the kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’. Both C and Gamma values can be modified to come up with the better prediction result.

After implementing all the machine learning algorithms that are mentioned above, the grid search method also will be used to tune up the prediction score by changing the values of parameters.

Benchmark

The final model that we select for the project will be based on the F1 scores and the time complexity. The F1 score has a range between 0 to 1. The higher the F1 score the better the prediction. In order to obtain a benchmark value, we have generated extra machine learning algorithms and recorded the average F1 score. The extra algorithms that we used for comparisons are SGD Classifier, Gradient Boosting Classifier, and One Vs Rest Classifier. The average F1 score ranges between 0.85 to 1.00. Intuitively, we can say that the prediction accuracy is high if its score above 80%. Thus, we can conclude that if an F1 score of the model that falls between 0.85 - 1.00 is a sufficient prediction for the project.

Moreover, we need to change the different training size and set various conditions on several features to measure each algorithm's performance to avoid overfitting. The overfitting algorithm can not be used to generalize the result of the prediction since the model is useless if we have different data set.

Methodology

Data Preprocessing

The target feature for the algorithm to make the prediction is CDR. Four different values are given in the feature to represent different levels of dementia. However, we want to predict whether the subjects has Alzheimer Disease. Thus, we need to convert those values into binary numbers, either 0 or 1. The 0 represents the subject does not have dementia, 1 represents the subject have dementia. We can use these binary CDR values to train the machine learning algorithms to make the prediction.

Furthermore, the gender feature M/F has non-numeric values. We should also convert the values into binary numbers so that the algorithms can use the gender feature. The ID and Hand features also have non-numeric values. However, the features are useless since ID is not necessary for predicting Alzheimer Disease and all the subjects in the data set are right handed. As we mentioned the reason earlier, the Delay feature is also not necessary for the project. Thus, we should drop the Delay, ID and Hand features.

At last, the data set has many empty values. Many of the subjects have missing information on the data. If we delete all the subjects who have missing data, the data set will lose its quality. Thus, we should use the mean value to fill in the empty table since this is the most representative values that we can assume. However, the mean will sometimes give us float values that can not be used in the certain feature. For instance, float values are not supported in SES feature because it only compares the educational level with integers. Thus, for these features, we should round the values into integers.

# The following code is the example of the procedures that we mentioned above.
First, we need to drop ID, Hand, and Delay features and create a new data since we are not going to use these three features in the project.

```
# Assign the data values into a new varaible and drop the features that is not necessary for the project
full_new_data = data.drop(["Delay","Hand","ID"], axis = 1)
```

Then, we need to convert the CDR values to binary values so that the machine learning algorithm can “learn” to classify the result. We should use python map method to convert the CDR values. If CDR > 0, then convert the values to 1. Otherwise, set it to 0.

```
# Use python map method to convert the CDR values. If CDR >= 0.5 convert the values to 1. Otherwise, set it to 0
full_new_data['CDR'] = full_new_data['CDR'].map({0.5: 1, 1: 1, 2:1, 0.3:0, 0:0})
```

Also, We have many empty NaN values in the dataset. It could create a problem when we use the supervised learning algorithm to train the data. Therefore, we need to use pandas library function to replace those NaN values with the mean value.

```
# Test how many columns have null value
data.columns[pd.isnull(data).any()].tolist()
```

```
# We have many empty NaN values in the dataset. Replace those values with mean value.
full_new_data = full_new_data.where(pd.notnull(full_new_data), full_new_data.mean(), axis='columns')
```

Since some of the features do not support float values after we fill in the mean value, we need to round the float values to integers.

```
# Round the float value into integer since some of the features do not support float values
full_new_data['SES'] = full_new_data['SES'].round(0)
```

Implementation

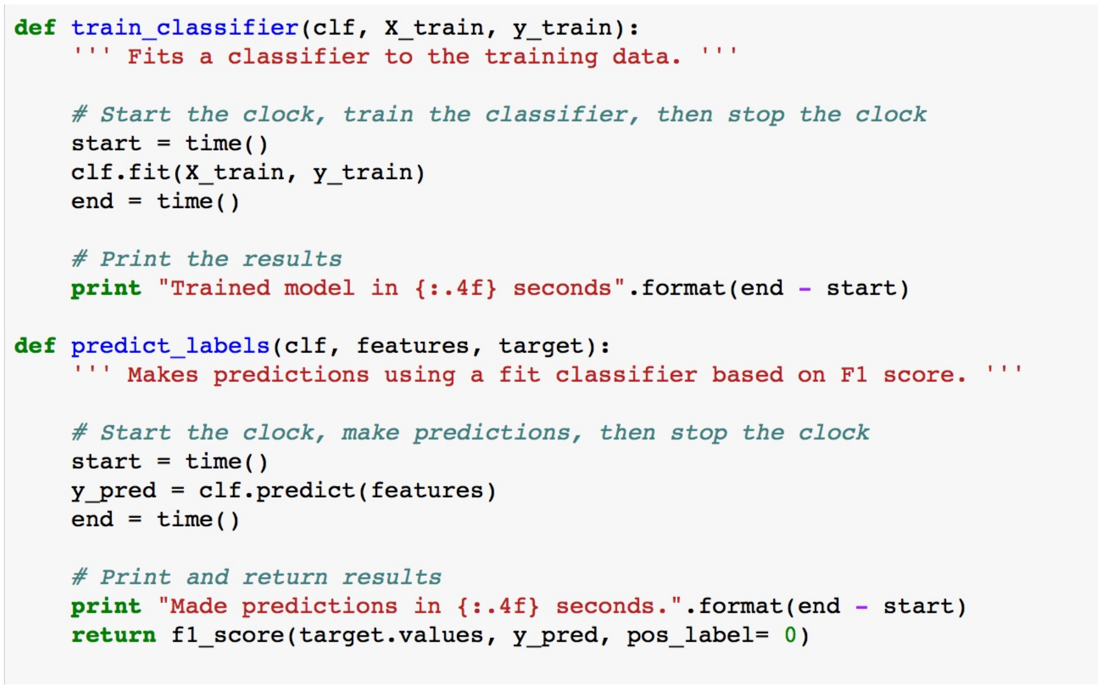
After the modification of the data set, we can implement machine learning algorithm to make the predictions. The process of implementing the three machine learning algorithms are relatively less complicated than the data processing.

We first shuffle and split the data into training and testing set.

```
# Import the train test split function from sklearn
from sklearn.cross_validation import train_test_split

# Shuffle and split the dataset into the number of training and testing
X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, train_size = 0.5)

# Show the results of the split
print "Training set has {} samples.".format(X_train.shape[0])
print "Testing set has {} samples.".format(X_test.shape[0])
```

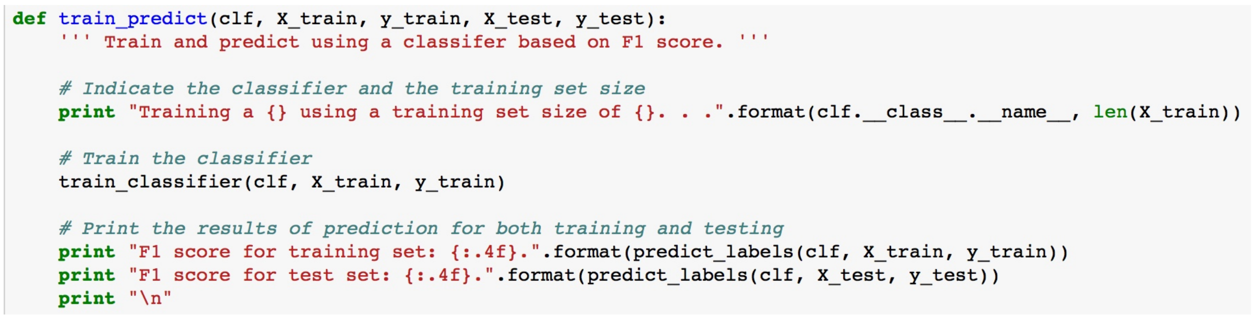
```
Training set has 218 samples.
Testing set has 218 samples.
```

Then, we created train_classifier and predict_labels function to print and record the running time of training and predictions.



```
def train_classifier(clf, X_train, y_train):
    ''' Fits a classifier to the training data. '''

    # Start the clock, train the classifier, then stop the clock
    start = time()
    clf.fit(X_train, y_train)
    end = time()

    # Print the results
    print "Trained model in {:.4f} seconds".format(end - start)

def predict_labels(clf, features, target):
    ''' Makes predictions using a fit classifier based on F1 score. '''

    # Start the clock, make predictions, then stop the clock
    start = time()
    y_pred = clf.predict(features)
    end = time()

    # Print and return results
    print "Made predictions in {:.4f} seconds.".format(end - start)
    return f1_score(target.values, y_pred, pos_label= 0)
```

Also, we implemented the train_predict function to show the F1 score of training set and testing set.



```
def train_predict(clf, X_train, y_train, X_test, y_test):
    ''' Train and predict using a classifer based on F1 score. '''

    # Indicate the classifier and the training set size
    print "Training a {} using a training set size of {}. . .".format(clf.__class__.__name__, len(X_train))

    # Train the classifier
    train_classifier(clf, X_train, y_train)

    # Print the results of prediction for both training and testing
    print "F1 score for training set: {:.4f}.".format(predict_labels(clf, X_train, y_train))
    print "F1 score for test set: {:.4f}.".format(predict_labels(clf, X_test, y_test))
    print "\n"
```

At last, we imported the Gaussian Naive Bays, Decision Trees Classifier and Support Vector Machines models from sklearn and initialized each of the three models. Then, we executed the train_predict function for each classifier and observed results of the predictions.

```
# Import the three supervised learning models and F1 score from sklearn
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn import tree
from time import time
from sklearn.metrics import f1_score

# Initialize the three models
clf_A = GaussianNB()
clf_B = tree.DecisionTreeClassifier(max_depth = 1000, min_samples_split = 10, random_state = 42)
clf_C = SVC(C=1000, kernel='rbf', gamma= 1e-4, random_state = 42)

# Execute the 'train_predict' function for each classifier and each training set size
for clf in [clf_A, clf_B, clf_C]:
    for i in [10,20,30,40,50,60,70,80,90,100,120,140,160,180,200]:
        train_predict(clf, X_train[:i], y_train[:i], X_test, y_test)
    print "\n"
```

The complicated part of the supervised learning implementation is that we do not know which values of parameters give us the fittest results for the data set. Thus, we need to constantly change each parameter to compare the results with the F1 scores that we mentioned in the benchmark section. The implementations are not difficult; rather, it is a time-consuming process to test each parameter of the algorithms. Furthermore, the prediction results are not the only interest for our project. When we choose the final machine learning model, we need to also conscious of not using an algorithm that gives us overfitting result because we cannot use the overfitting model to generalize the prediction.

Refinement

The training size was set to 10, 30, 50, 100 and 218 to test each of the algorithms. The Gaussian Naive Bays algorithm almost always give the perfect F1 score. Even when the training size is very small, the algorithm still shows the 1.0 F1 score. Similarly, the Decision Tree Classifier also shows the same results. Although we set the parameters max_depth and min_samples_split with the ridiculous values, the prediction result remains at the almost perfect F1 score as the training size increases to 218. Even if the training size is as small as 10, the Decision Tree still manages to have the F1 score of 0.9333 which is very strange.

Unlike Gaussian Naive Bays and Decision Tree Classifier, Support Vector Machine shows the most reasonable result. Initially, we used default parameters to test the SVM, and the results were quite interesting. When the training size was set to 10, the SVM F1 score for the test was 0.7092. Then, as the training size increased to 218, the test score also increases to 0.8779. After that, we modified the parameters values and set C=1, kernel='sigmoid', gamma= 0.1 then retest the SVM. The final score was increased to 0.8906. Furthermore, we change the kernel method to "rbf" again and retest the SVM. Interestingly, the result has a minor improvement. It shows the F1 score of 0.8929.

If we observe the result, we can make the conclusion that GNB and DT may have overfitting result. The results of the Gaussian Naive Bays and Decision tree are too good to be true. If we have different data set, these two models that we implemented would give the poor predictions. Although SVM has the lowest score among the three algorithms, it is important to know that SVM avoids overfitting. Thus, it is the best model among the three algorithms for predicting Alzheimer Disease.

Results

Model Evaluation and Validation

Since one of the advantages of SVM is avoiding overfitting, we select the SVM as our final model to predict dementia. The parameters that we use to test the algorithms are gamma, C, and kernel. The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. The C parameter trades off misclassification of training examples against the simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors. We gradually increase the values of C and lower the values of gamma to improve the accuracy of the SVM predictions. The final F1 score for test set is 0.9056 when we use parameters C = 1000, gamma = 0.01 and kernel='rbf'. Also, the changes of training size greatly influence the results of SVM as well. The F1 test score for train size of 10, 30, 50, 100 and 218 are 0.8918, 0.8918, 0.9027, 0.8833 and 0.9056.

The parameters are not the only modification that we made for testing the algorithm. We also used Grid Search method to tune up the result of the SVM. By setting the kernel = 'rbf', gamma = [1e-3, 1e-4], C = [1, 10, 100, 1000], we successfully increased the F1 score to 0.9687 which is almost closed to the perfect prediction score. The training size and test size are split equally in the final model and the score 0.9687 is robust enough to make the prediction of Alzheimer Disease.

The only disadvantage of the SVM is that the time complexity significantly increases when the data size increases. The Gaussian Naive Bays and the Decision Tree Classifier algorithms' training time ranges between 0.0002 seconds to 0.0013 seconds. In contrast, the SVM's running time increases from 0.0004 seconds to 0.0046 seconds as we increase the training size. Luckily, the data only includes 436 samples so it does not make too many differences when we train the model. However, if we have a huge data set, the SVM model could be a very accurate but an inefficient algorithm to make predictions.

Justification

The final SVM model has the prediction score of 96.87% which falls between the benchmark range 0.85 - 1.00, and the exponentially growing running time does not significantly influence the efficiency with the OASIS data set.

The SVM model does not have the perfect score even after applying the Grid Search method and the time complexity is the highest among the three algorithms. However, another important hypothesis that we made in the benchmark section is that we want to choose the algorithm that has no overfitting. Thus, we can conclude that the SVM is the best model among the three algorithms to predict the Alzheimer Disease with the OASIS data set.

Conclusion

Visualization

To track the improvement of the prediction, we started with the training size of 2 and gradually increased the training size to 200. Then we recorded each algorithm's F1 test score correspond to the training size.

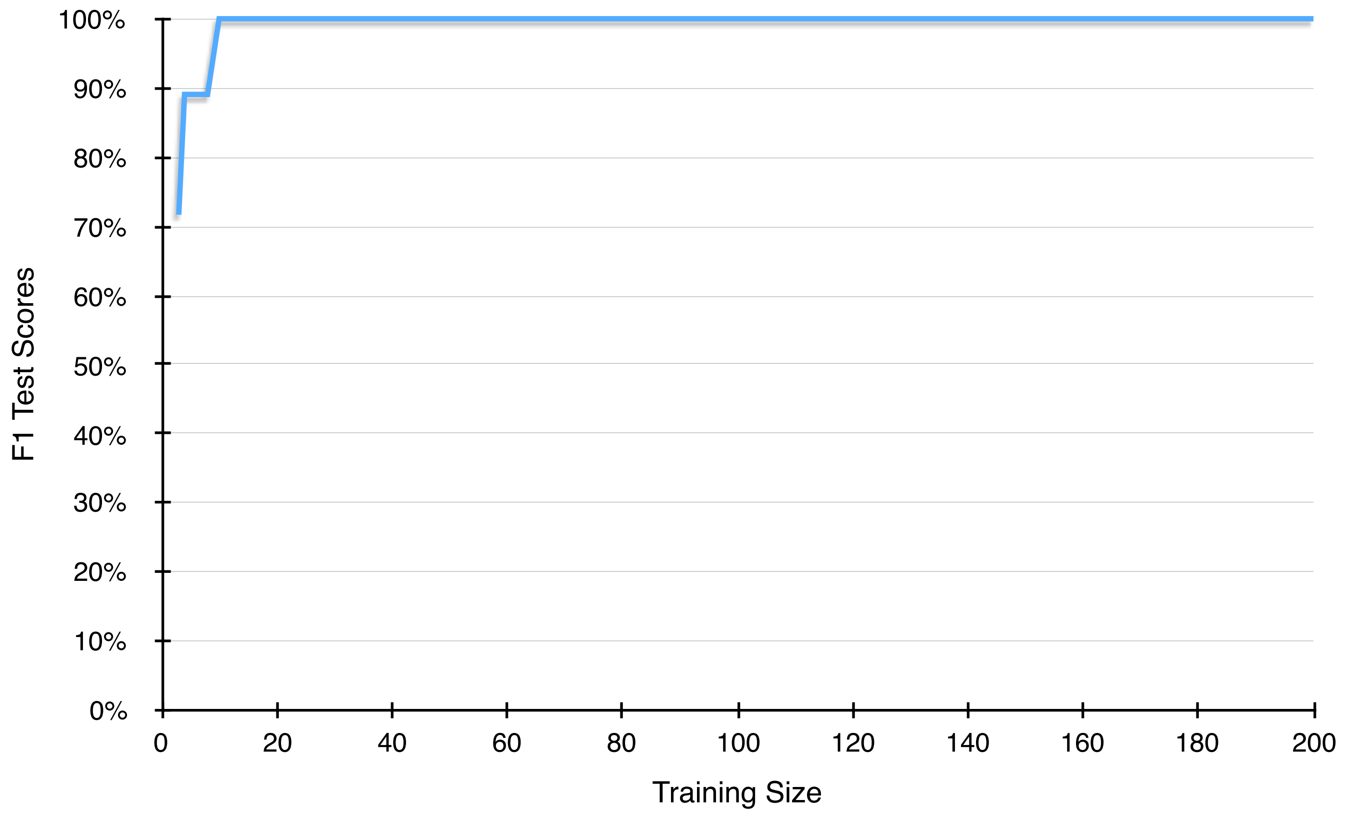


Figure 3. Gaussian Naive Bays Prediction Scores and Training Size

As we can see from the Figure 3, GNB has very high prediction score. Even when the training size is as small as 2, the GNB already has approximately 72% prediction accuracy. When the training size reaches 10, the GNB has obtained the perfect F1 score. As the training size keep increasing, the GNB maintains the 100% prediction score.

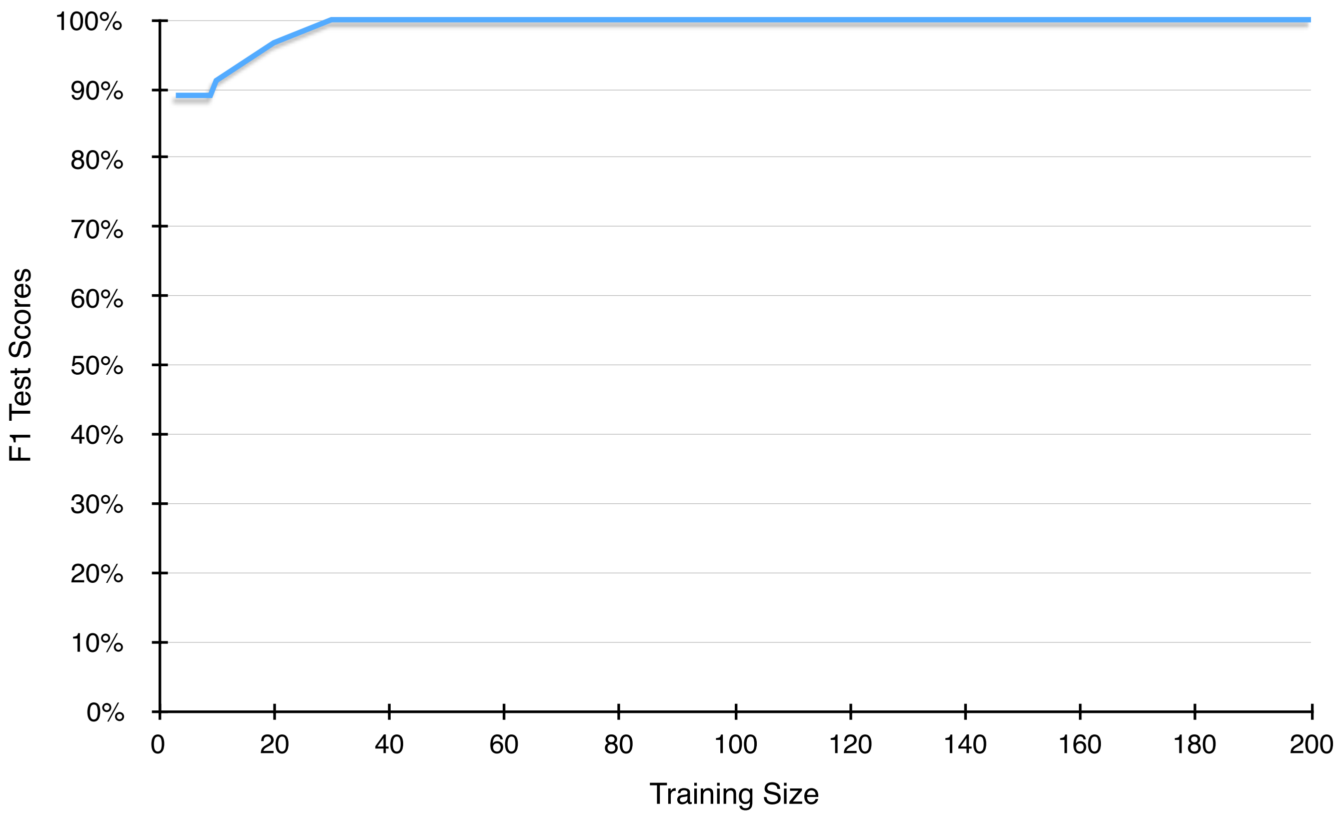


Figure 4. Decision Trees Classifier Prediction Scores and Training Size

More surprisingly, the DT has better performance than the GNB with the OASIS data set. When the training size is 2, the DT has already reached 90% prediction score. Just as the GNB, the F1 score of DT also increases as the training size keep increasing, and it maintains the perfect score until the algorithm reached the training size 200.

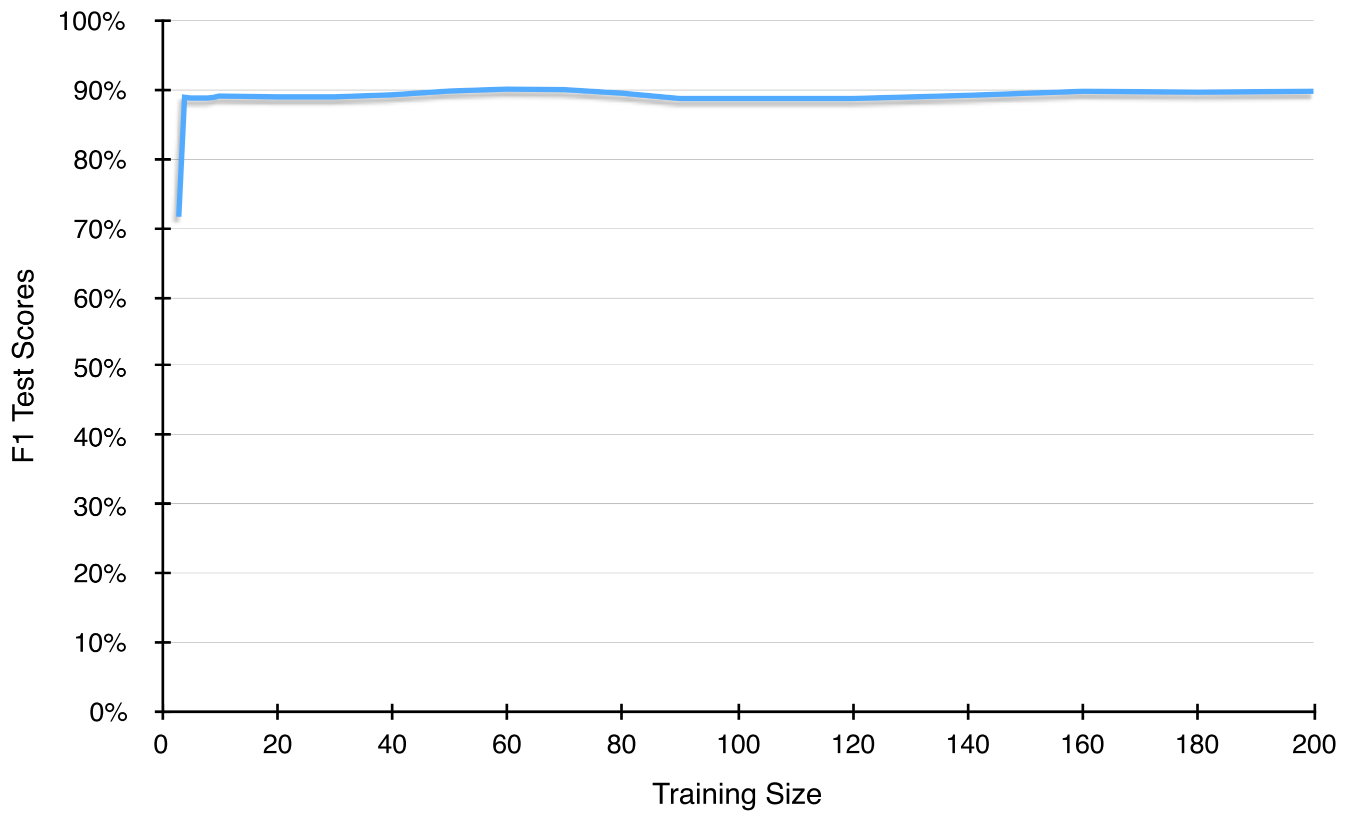


Figure 5. Support Vector Machines Prediction Scores and Training Size(Before Grid Search)

Unlike the DT and the GNB, SVM has relatively low prediction score during the testing. Before used the Grid Search method, SVM started with the approximately 73% prediction score and maintains the scores between 87% - 89%. It sometimes reaches 90%, but it is not always the case.

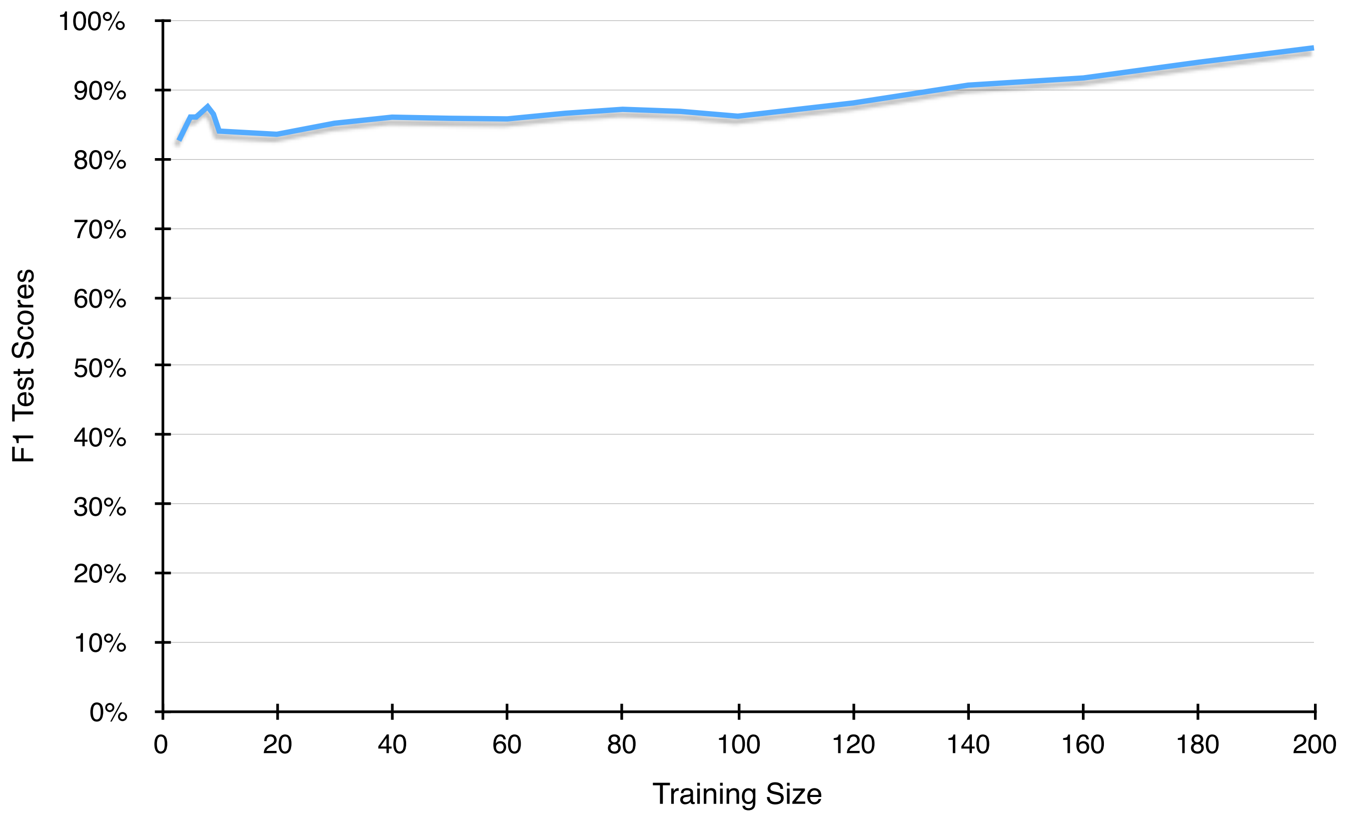


Figure 6. Support Vector Machines Prediction Scores and Training Size(After Grid Search)

After using the Grid Search method, the SVM finally increased the overall performance. However, when we compare the score in figure 6 with the score in figure 5, we can notice that after applying the Grid Search, the score is even lower than before we tuned up the algorithm until it reaches the training size 140. After that, as the training size increases, the prediction score simultaneously increases.

Reflection

The initial approach of predicting Alzheimer Disease was completely different. As we mentioned in the beginning of the project, we need to first solve the problems of missing information in the data set. If we keep NaN values in the data set, we would not be able to implement the machine learning algorithms to predict dementia. Thus, we initially deleted all the subjects who have empty values in certain features instead of filling the mean values. The problem with this approach is obvious. The sample size reduced dramatically from 463 to 236. Almost half of the data information was deleted and the data lost its quality. When we used the reduced data to examine the machine learning algorithms, all three of the algorithms got almost 100% prediction scores. This is the result that we cannot use to generalize any similar problems. Thus, we changed our approach.

Instead of simply deleting the information from the data set, we filled in the mean values to the OASIS data. This approach also causes certain problems. By filling in the mean values, we might have set the people who did not have dementia falsely classified to be demented. However, we did not miss overall quality of the data set by doing this approach. We still maintained the sample size and our assumption did not completely twist the data information. Thus, we were able to use the complete samples to make our general prediction.

After we had filled in the mean values, we needed to convert Clinical Dementia Rating values into binary values to do the classification. The conversions of CDR values were critical for our algorithms to make predictions. We used the map function to convert the CDR values in the project. However, it is not an efficient approach and it could potentially raise a problem. If we want to convert the CDR values that is greater than 0.5 to 1, we will have trouble to implement it by using map function. For instance, if we want to convert 0.501,0.502,0.503…0.5011,0.5012…, we would have to map all these values to 1 which is extremely inefficient. We choose to use map function in the project because the data set only has four values which is easy to convert. However, in general, it is not always the best approach to solving similar problems.

The process of implementing the machine learning algorithms was not challenging. We mostly used the examples from scikitlearn documentations. The only modification that we made was the change of parameters in each algorithm. Among the three algorithms, SVM had the fittest result in our data set even though it did not have the highest prediction scores. The SVM model that was implemented in the project is the only model that can be generalized to solve other similar types of problems.

Improvement

The SVM model has the prediction accuracy of 96% which is very high. However, the time complexity is still a problem if the data set gets large. As I research different classification machine learning algorithms, I found out that Random Forest Classifier algorithm could be useful to solve our problem. The Random Forest Classifier is a meta-estimator that fits some decision tree classifiers on various subsamples of the dataset. It uses averaging to improve the predictive accuracy and control overfitting. If we have a larger data set and want to avoid overfitting result, I think Random Forest Classifier would be a good choice.

References

"Alzheimer's Disease & Dementia | Alzheimer's Association." Alzheimer's Disease & Dementia | Alzheimer's Association. N.p., n.d. Web. 24 Jan. 2017.

"Alzheimer's & Dementia Prevention and Risk | Research Center." Alzheimer's Association. N.p., n.d. Web. 25 Jan. 2017.

"ETIV - estimated Total Intracranial Volume, aka ICV." ETIV - Free Surfer Wiki. N.p., n.d. Web. 27 Jan. 2017.

"Support Vector Machines" Support Vector Machines — scikit-learn 0.18.1 documentation. N.p., n.d. Web. 01 Feb. 2017.

"Sklearn.svm.SVC." Sklearn.svm.SVC — scikit-learn 0.18.1 documentation. N.p., n.d. Web. 01 Feb. 2017.

"RBF SVM parameters." RBF SVM parameters — scikit-learn 0.18.1 documentation. N.p., n.d. Web. 02 Feb. 2017.

"RandomForestClassifier." sklearn.ensemble.RandomForestClassifier — scikit-learn 0.18.1 documentation. N.p., n.d. Web. 02 Feb. 2017.