Course Name: - Applied Data Science

Course Number: - 6300

Semester: - Fall 2022

Semester: - 001

Autism Spectrum Disease Biolog

Venkata Sai Pratyusha Samidhapudi	Sai Vipraj Telukoti	Chandana Pothugunta	Bharath Chittamuru
vsamidh@clemson.edu	steluko@clemson.edu	cpothug@clemson.edu	bchitta@clemson.edu
EDA	Decision Tree	SVM with radial kernel	Gradient Boosting Classifier
Random Forest Classifier	SVM with Linear Kernel	Summary and Scatter Plot	Introduction
Summary of Machine Learning models	Summary of EDA.	Conclusion	Scatter Plot

1. INTRODUCTION

What is the main question your project seeks to answer?

Autism is a disorder that is related to complex neurobehavior. It is characterized by the impairment in reciprocal social interaction, impairment in communication, and the presence of repetitive and stereotypical patterns of behaviors, interests, and activities. Autism Spectrum Disorder is a single disorder described in Diagnostic and Statistical Manual of Mental Disorders (DSM-5) which encompasses AD (Autistic Disorder), PDD-NOS (Pervasive Developmental Delay-Not Otherwise Specified), and Asperger syndrome. The onset of symptoms is typically before the age of 3 years. Autism spectrum disorders are estimated to occur in as many as 1: 59 (one in every 59 children). The motive of the study is to explore the metabolic profiles of cell lines from the cohorts and identify significant differences and similarities, verify if ASD cells are more distinguishable from controls or if they overlap for certain key metabolic traits. Comparing the controls and ASD patients' profiles to identify if there are metabolic differences that are consistent enough to be proposed as biomarkers.

Provide a brief motivation for your project question. Why is this question important? What can we learn from your project?

Diagnosing ASD can be demanding since there is no specific medical test, like a blood test, to diagnose the disorder. The current diagnosing time is a huge challenge for Autism. It takes up to six months to firmly diagnose a child with autism due the lengthy procedure, and a child must visit different specialists to diagnose autism, starting from developmental pediatricians, neurologists, psychiatrists, or psychologists. As a diagnosis procedure, Doctors analyze the child's behavior and development. ASD can sometimes be detected at 18 months of age or younger. By age 2, a diagnosis by an experienced professional can be considered reliable. However, many children do not receive a final diagnosis until they are much older. Some people are not diagnosed until they are adolescents or adults. This delay means that people with ASD might not get the early help they need. Diagnosing ASD in adults is often more difficult than diagnosing ASD in children. In adults, some ASD symptoms can overlap with the symptoms of other mental health disorders, such as anxiety disorder or attention-deficit/hyperactivity disorder (ADHD).

The abstruseness of the diagnostic process for autism spectrum disorder makes it an interesting topic for the development of a deep learning application and for two reasons:

- To utilize a massive amount of health and medical data available towards predictive modeling and predictive analysis
- Opportunity for clinical aid by providing higher accuracy in correctly diagnosing autism spectrum disorder by observing the metabolic traits.

Briefly describe the data source(s) you have used in your project. Where is the data from? How big is the data in terms of data points and/or file size? If the data was not already available, how did you collect the data?

The source of the dataset is Greenwood Genetic Center. The dataset is gathered from the experiments conducted on Lymphoblastoid Cell Lines (LCLs) using the Biolog Phenotype MicroArrays (PM) which are preconfigured 96-well microplates coated with different oxidizable carbon sources in various wells. The dataset is organized into 2 different folders namely, Controls and ASD Patients. There are subfolders in each of them that corresponds to 8 Phenotype MicroArrays (PM) and Tryptophan plate.

2. SUMMARY OF EDA

What is the unit of analysis?

The motive of this model is to explore the metabolic profiles of cell lines from the different cohorts (namely Controls, ADS) to find any significant similarities or differences between them. Also, to verify if there is any overlap in the metabolic traits between AD or ASD. There are 50 people in each cohort. We have calculated average for all the metabolic profiles in each plate for controls and ASD patients. The averages from Controls and ASD patients are used to make comparisons, in order to identify any overlap between metabolic traits.

How many observations in total are in the data set?

There are metabolic profile details for 50 people from each cohort. The data is collected across PM-M1 to PM-M8 and TRP plate, which gives a total of 9 plates.

How many unique observations are in the data set?

All the observations are collected from different patients and as the data is metabolic profiles of the individuals, they are unique.

What time period is covered?

The time period covered in our dataset was for 4 years.

Briefly summarize any data cleaning steps you have performed

The rows “blood sample date” and “assay date” are not related to the analysis. Hence, we have excluded the unwanted rows. The column named ‘well’ plays no crucial role while analyzing the metabolic profiles, so it was dropped. The indices were reset, and we added a new column “Average" which is the mean value of each metabolic profiles of all the patients.

```
pm2=pd.read_excel("C:/Users/14708/OneDrive/Documents/ADS/ASD Biolog Project - Raw Data/50 ASD/ASD PM
pm2 = pm2.drop(0,axis=0)
pm2 = pm2.drop('A1',axis=1)
pm2=pm2.rename(columns=pm2.iloc[0]).drop(pm2.index[0])
pm2 = pm2.reset_index(drop=True)
pm2['Average'] = pm2.loc[:, pm2.columns!='CMS#'].mean(axis=1)
pm2.head()
```

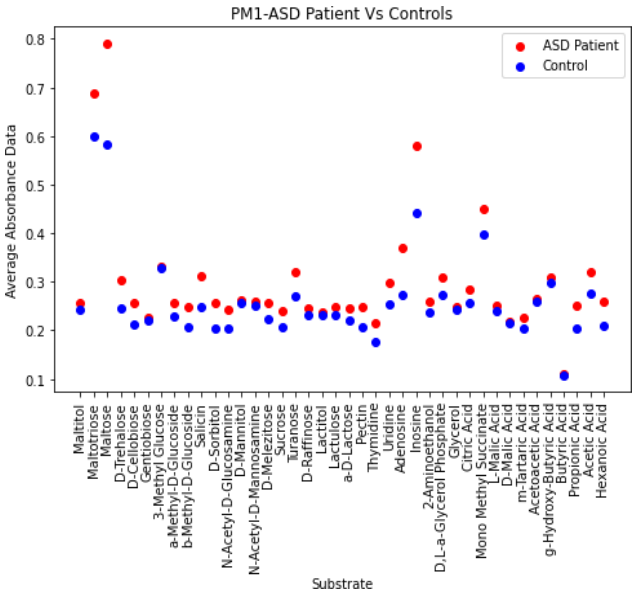
Visualization of the response with an appropriate technique

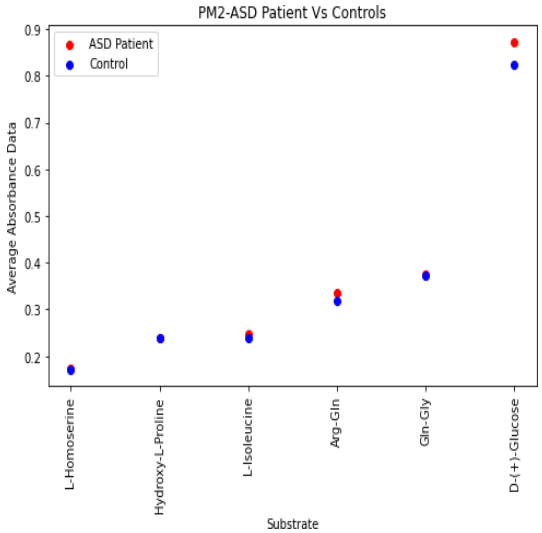
Algorithms: Support Vector Machine Classifier, Random Forest Classifier, Gradient Boosting Classifier, Decision Tree, and k-Nearest Neighbor Classifier.

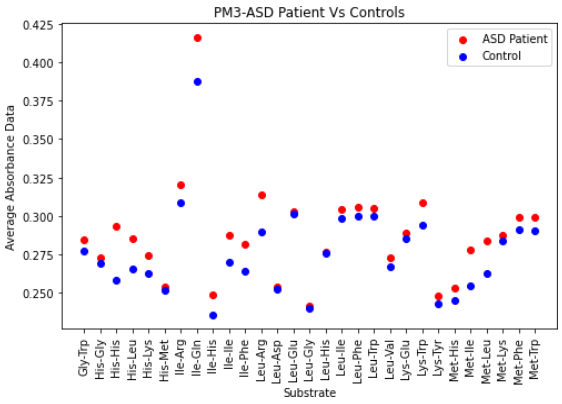
Visualization of key predictors against the response

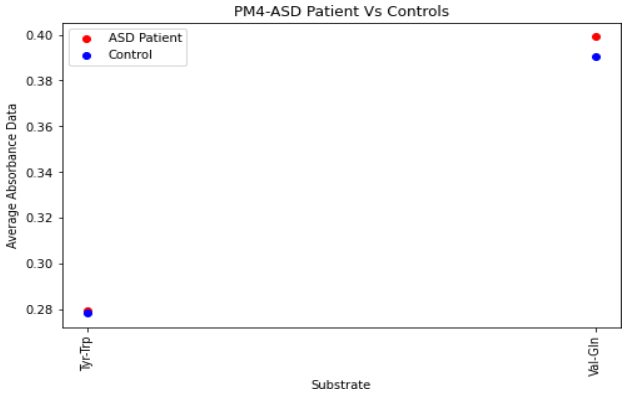
Detecting the significant differences in the metabolic profiles and points of solvents collected from the cohorts for 9 Phenotype MicroArrays (PM) plates. Comparison is done between the average metabolic data of 50 ASD Patients’ and 50 Controls. In addition, this will also experiment with threshold values to distinguish the cohorts from themselves. After incorporating the data, the model we create, predicts which category the sample falls under with decent precision.

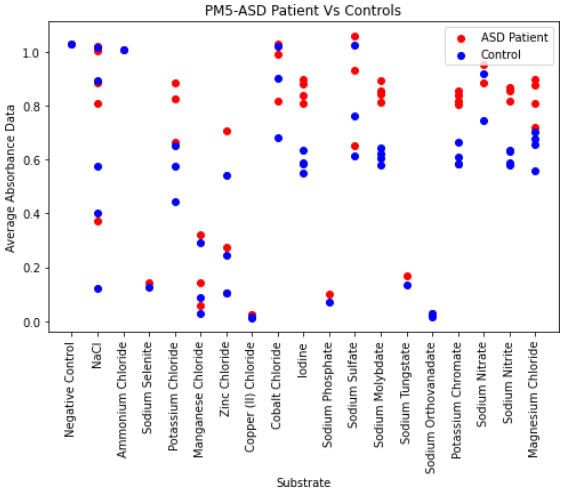
The following scatter plots visualize the metabolic profiles whose value is more for Controls than those of ASD Patients. These metabolic profiles can help distinguish the cohorts potentially.

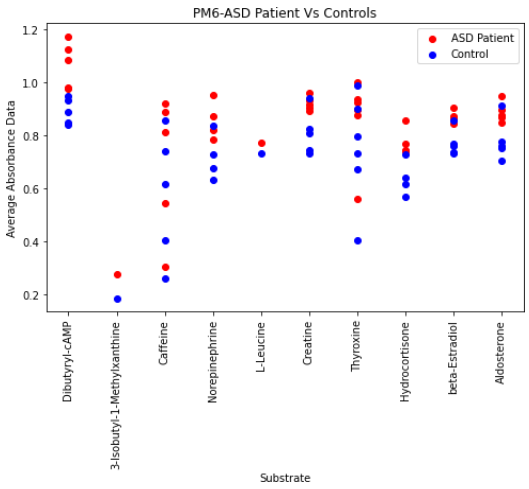


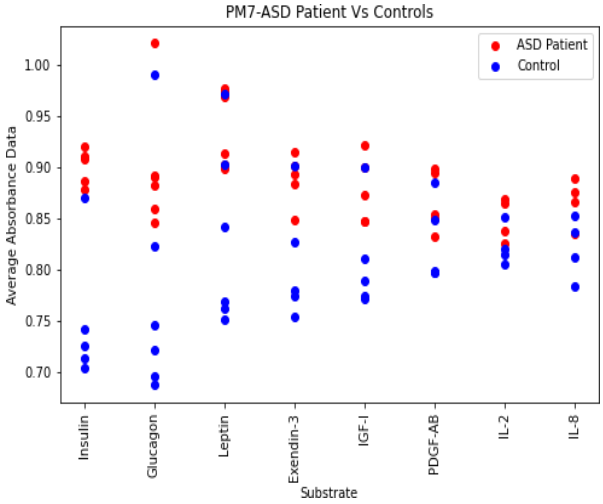


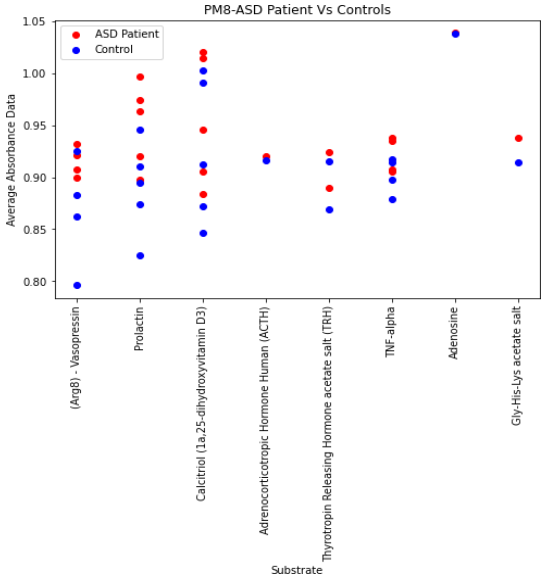

PM3-ASD Patient Vs Controls
ASD Patient
Control
Average Absorbance Data
Substrate
Gly-Trp
His-Gly
His-His
His-Leu
His-Lys
His-Met
Ile-Arg
Ile-Gln
Ile-His
Ile-Ile
Ile-Phe
Leu-Arg
Leu-Asp
Leu-Glu
Leu-Gly
Leu-His
Leu-Ile
Leu-Phe
Leu-Trp
Leu-Val
Lys-Glu
Lys-Trp
Lys-Tyr
Met-His
Met-Ile
Met-Leu
Met-Lys
Met-Phe
Met-Trp

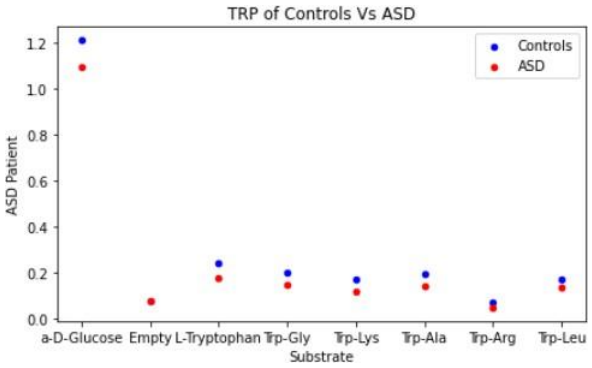

PM4-ASD Patient Vs Controls
ASD Patient
Control
Average Absorbance Data
Substrate
Tyr-Trp
Val-Gln


PM5-ASD Patient Vs Controls
ASD Patient
Control
Average Absorbance Data
Substrate
Negative Control
NaCl
Ammonium Chloride
Sodium Selenite
Potassium Chloride
Manganese Chloride
Zinc Chloride
Copper (II) Chloride
Cobalt Chloride
Iodine
Sodium Phosphate
Sodium Sulfate
Sodium Molybdate
Sodium Tungstate
Sodium Orthovanadate
Potassium Chromate
Sodium Nitrate
Sodium Nitrite
Magnesium Chloride

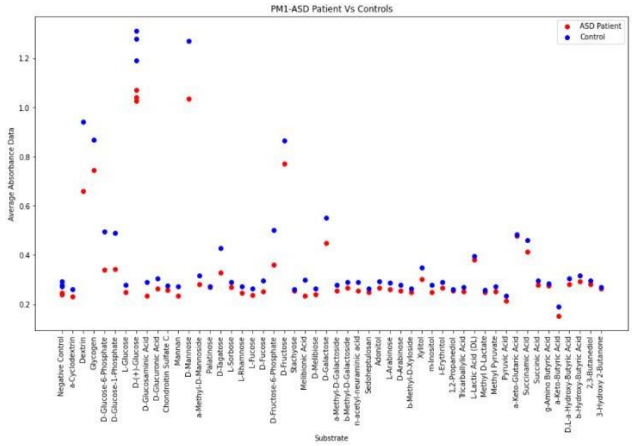

PM6-ASD Patient Vs Controls
ASD Patient
Control
Average Absorbance Data
Substrate
Dibutyryl-cAMP
3-Isobutyl-1-Methylxanthine
Caffeine
Norepinephrine
L-Leucine
Creatine
Thyroxine
Hydrocortisone
beta-Estradiol
Aldosterone

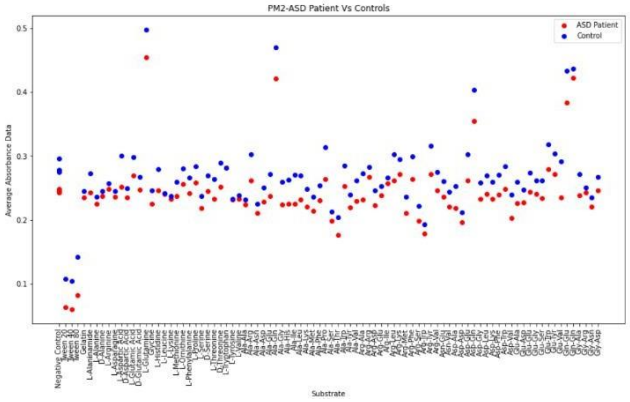

PM7-ASD Patient Vs Controls
ASD Patient
Control
Average Absorbance Data
Substrate
Insulin
Glucagon
Leptin
Exendin-3
IGF-I
PDGF-AB
IL-2
IL-8

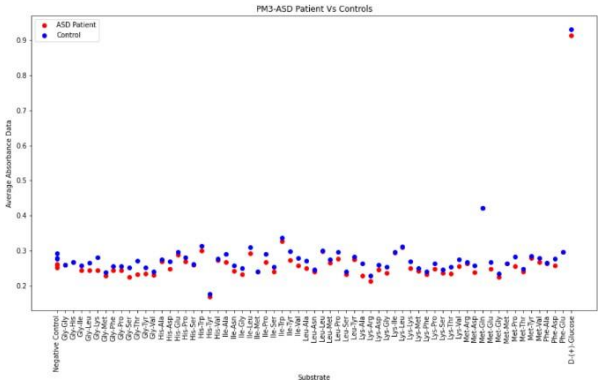

PM8-ASD Patient Vs Controls
ASD Patient
Control
Average Absorbance Data
Substrate
(Arg8) - Vasopressin
Prolactin
Calcitriol (1a,25-dihydroxyvitamin D3)
Adrenocorticotropic Hormone Human (ACTH)
Thyrotropin Releasing Hormone acetate salt (TRH)
TNF-alpha
Adenosine
Gly-His-Lys acetate salt

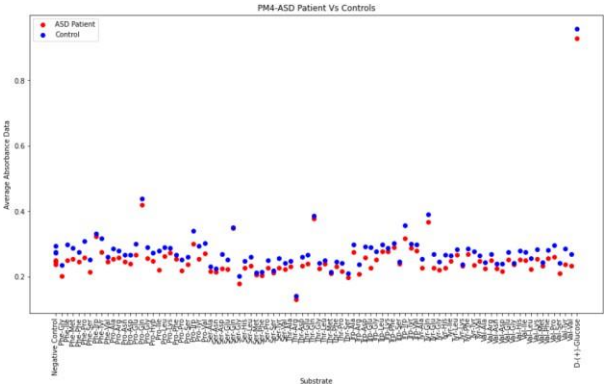


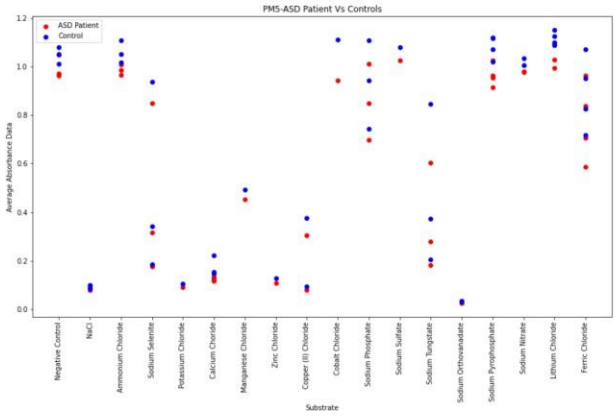
To take all the metabolic profiles into consideration and ensure that we do not miss out on utilizing the data to predict with precision, we have visualized the substrate whose value is more for ASD patients than those of Controls.

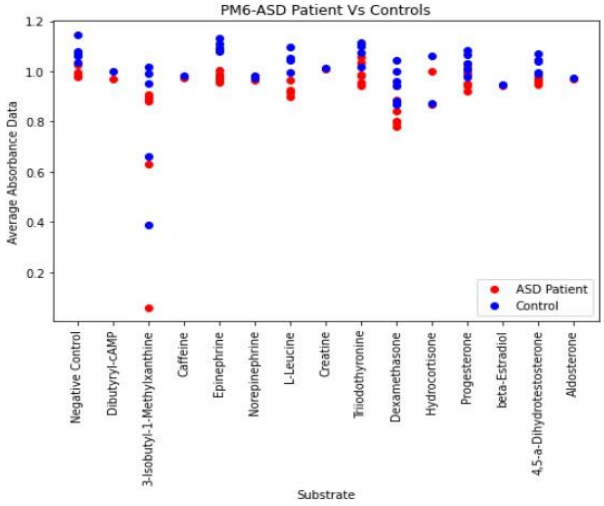


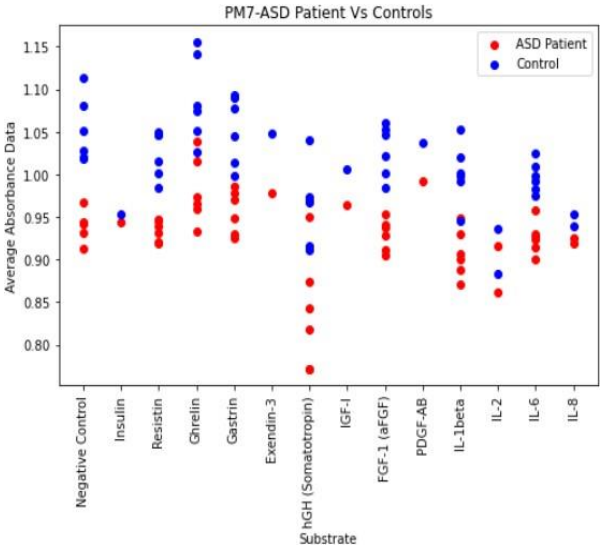


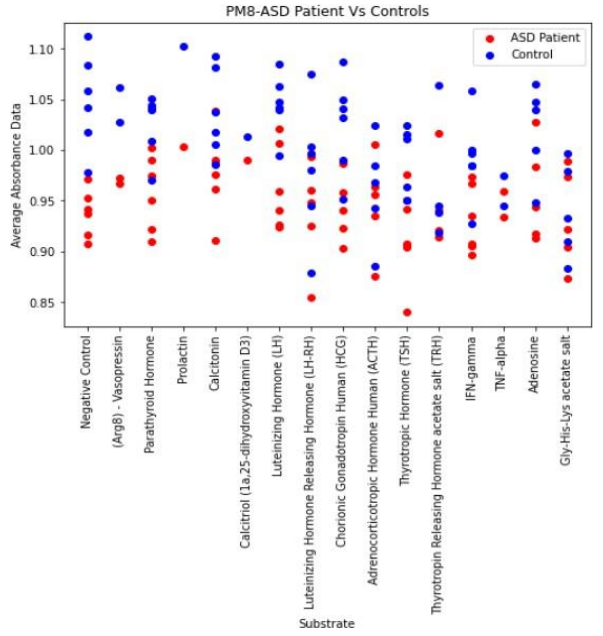


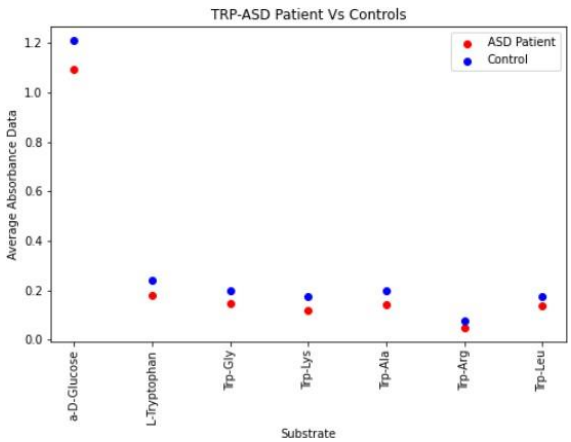












3. SUMMARY OF MACHINE LEARNING MODELS

The dataset has been divided into testing and training data. As per the observations drawn from EDA, we have concluded that using classifier algorithms like Linear Support Vector Machine (SVM), Random Forest Classifier, Gradient Boosting Classifier, Decision Tree, and k-Nearest Neighbor Classifier would lead better results. To choose among these models which best fits the dataset, we made comparisons of various metrics like prediction accuracy.

Justify your model choices based on how your response is measured and any observations you have made in your EDA.

We have chosen to work with Gradient Boosting classifier for the following reasons:

- Often achieves predictive accuracy that cannot be trumped.
- Flexibility - can optimize on different loss functions and provides several hyper parameter tuning options that make any function fit flexibly.
- No data pre-processing required - often works great with categorical and numerical values as is.

The reason we thought Random Forest Classifier works well with our dataset is because a huge number of relatively uncorrelated trees operating as a group will outperform any individual constituent models. Since it also known to perform better than the SVM as it is much suited for multiclass problems where SVM is suited for two-class problems, Random Forest Classifier seemed like a much better choice.

k-Nearest Neighbor Classifier needs no assumptions about the data, tune several parameters or build a model. This seemed like a reasonably good choice since it can make highly accurate predictions, but it doesn't require human-readable model.

Report the results from at least two different models:

For each model, report the model's test error. Justify your choice.

For each model, discuss how well the model fits the data.

The following metrics were calculated for each model:

- F1-Score: - It sums up the predictive performance of a model by combiningtwo otherwise competing metrics — precision and recall.

- Confusion matrix: - It helps in identifying which classes are easy to predict and which are hard to predict. It provides how many examples for each class are correctly classified and how many are confused with other classes. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class.
- Recall: - Recall is a metric that quantifies the number of correct positive predictions made from all positive predictions that could have been made. Unlike precision that only comments on the correct positive predictions out of all positive predictions, recall provides an indication of missed positive predictions.
- Accuracy: - It is a way to measure how often the algorithm classifies a datapoint correctly. Accuracy is the number of correctly predicted data points among all the data points.
- Precision: - Precision is one indicator of the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions.

1. Random Forest Classifier:

After test and train split on the data, we have applied Random Forest Classifier. Since, it generates reasonably good predictions across wide range of data while requiring little configuration. This classifier constructs multitude of Decision trees at the training time and the output for the presented classification problem is the one which is selected by the most number of trees.

The prerequisites for random forest to perform well are: There needs to be some actual signal in our features so that models built using those features do better than random guessing; The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.

The code snippet shown here represents the implementation of Random Forest Classifier and the related metrics. The Accuracy is around 82% which indeed means reliable model performance. One way to test the error rate is by using the Mean Absolute Error formula. To determine how well this model suits our dataset. This could ensure that there is a much less chance of inaccurate prediction and that is a minimal chance of overfitting. The Mean Absolute Error is around 0.17 which is a pretty decent value, and it proves that this classifier fits with the data.

```
#Random Forest Classifier
import seaborn as sb
from sklearn import ensemble

rf_clf=ensemble.RandomForestClassifier(n_estimators=100)
rf_clf.fit(X_train,y_train)
print("Accuracy on training set:",rf_clf.score(X_train,y_train))
print("Accuracy on test set:",rf_clf.score(X_test,y_test))
rfy_pred=rf_clf.predict(X_test)

print("Classification report:")
print(classification_report(y_test,rfy_pred))
print(metrics.accuracy_score(y_test,rfy_pred))
```

```
Accuracy on training set: 1.0
Accuracy on test set: 0.8205128205128205
Classification report:
              precision    recall  f1-score   support

           1       0.74      0.88      0.80        16
           2       0.90      0.78      0.84        23

    accuracy                           0.82        39
   macro avg       0.82      0.83      0.82        39
weighted avg       0.83      0.82      0.82        39

0.8205128205128205
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
#define cross-validation method to use
print(cross_val_score(rf_clf, X, y, scoring='accuracy',cv=10).mean())
```

```
0.7563157894736843
```

```
print('F1 : ', f1_score(y_test, rfy_pred, average=None))
print('Precision : ',precision_score(y_test, rfy_pred, average=None))
print('Recall : ', recall_score(y_test,rfy_pred, average=None))
```

```
F1 :  [0.8       0.8372093]
Precision :  [0.73684211 0.9       ]
Recall :  [0.875     0.7826087]
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, rfy_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, rfy_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, rfy_pred)))
```

```
Mean Absolute Error: 0.1794871794871795
Mean Squared Error: 0.1794871794871795
Root Mean Squared Error: 0.4236592728681617
```

2. Gradient Boosting Classifier:

It gives a prediction model which as a whole is an ensemble of weak prediction models, that are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. This model is usually built in a stage-wise fashion, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function. Since this method is known to stand out for its speed of prediction for large datasets in particular, this seems like a pretty interesting choice.

The code snippet shows the implementation of the Classifier and the calculation of all the metrics.

```
#Gradient Boosting Classifier
gb_clf=ensemble.GradientBoostingClassifier()
gb_clf.fit(X_train,y_train)
print(gb_clf.score(X_test,y_test))
gby_pred=gb_clf.predict(X_test)
print("Classification report:")
print(classification_report(y_test, gby_pred))
print("Accuracy on test set:",gb_clf.score(X_test,y_test))
print("Accuracy on train set:",gb_clf.score(X_train,y_train))
print(metrics.accuracy_score(y_test,gby_pred))

print("Confusion matrix:",confusion_matrix(y_test, gby_pred))
```

```
0.7692307692307693
Classification report:
              precision    recall  f1-score   support

           1       0.71      0.75      0.73        16
           2       0.82      0.78      0.80        23

    accuracy                           0.77        39
   macro avg       0.76      0.77      0.76        39
weighted avg       0.77      0.77      0.77        39

Accuracy on test set: 0.7692307692307693
Accuracy on train set: 1.0
0.7692307692307693
Confusion matrix: [[12  4]
 [ 5 18]]
```

```
print('F1 : ', f1_score(y_test, gby_pred, average=None))
print('Precision : ',precision_score(y_test, gby_pred, average=None))
print('Recall : ', recall_score(y_test,gby_pred, average=None))
```

```
F1 :  [0.72727273 0.8       ]
Precision :  [0.70588235 0.81818182]
Recall :  [0.75      0.7826087]
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, gby_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, gby_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, gby_pred)))
```

```
Mean Absolute Error: 0.23076923076923078
Mean Squared Error: 0.23076923076923078
Root Mean Squared Error: 0.4803844614152614
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
#define cross-validation method to use
print(cross_val_score(gb_clf, X, y, scoring='accuracy',cv=10).mean())
```

```
0.7460526315789473
```

The Mean Absolute Error is around 0.23 which reasonable value but is more than that of Random Forest Classifier. Which means that in comparison Random Forest Classifier performs much better.

3. Decision Tree:

By learning straightforward decision rules derived from training data, a Decision Tree creates a training model that may be used to predict the class or value of the target variable.

In decision trees, we begin at the tree's root when anticipating a record's class label. The root attribute's values will be compared. Decision trees are used to estimate the likelihood that various iterations of decisions will be successful in achieving a particular goal. Different nodes make up decision trees. The decision tree's root node, which in machine learning typically represents the entire dataset, is where it all begins. The leaf node is the branch's termination point or the final outcome of all the possible decisions. From a leaf node, the decision tree won't branch out any further. With decision trees in machine learning, the core nodes represent the data's features, and the leaf nodes represent the results.

```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)

tree_pred = tree.predict(X_test)
print(metrics.accuracy_score(y_test,tree_pred))
print("Accuracy on training set:",tree.score(X_train, y_train))
print("Accuracy on test set: ",tree.score(X_test, y_test))
```

```
0.8205128205128205
Accuracy on training set: 1.0
Accuracy on test set:  0.8205128205128205
```

```
print('F1 : ', f1_score(y_test, tree_pred, average=None))
print('Precision : ',precision_score(y_test, tree_pred, average=None))
print('Recall : ', recall_score(y_test, tree_pred, average=None))
```

```
F1 :  [0.81081081 0.82926829]
Precision :  [0.71428571 0.94444444]
Recall :  [0.9375     0.73913043]
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, tree_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, tree_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, tree_pred)))
```

```
Mean Absolute Error: 0.1794871794871795
Mean Squared Error: 0.1794871794871795
Root Mean Squared Error: 0.4236592728681617
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
#define cross-validation method to use
print(cross_val_score(tree, X, y, scoring='accuracy',cv=10).mean())
```

```
0.735
```

The accuracy is 82% which is significantly close to the performance of Random Forest Classifier, but after the cross validation the accuracy has been reduced by 0.02 in comparison to that of Random Forest Classifier.

4. K-Nearest Neighbor Classifier:

KNN is a technique for categorizing data points by comparing them to their nearest annotated data point, also referred to as closest neighbor.

KNN attempts to identify the group to which a data point belongs by examining the data points nearby. Let us consider two groups, A and B. The algorithm examines the nearby data points' states to determine if a data point belongs to group A or group B. It is quite likely that the data point in question is in group A if the bulk of the data points are in group A, and vice versa.

The figure below shows the implementation of k-NN Classifier. The accuracy on the test data is 74% which after the cross validation changed to 66% leaving much room for improvement. The Mean Absolute error is nearly 0.25. Among the 3 models that have been implemented Random Forest Classifier outplays all of them.

```
#kNN Classifier
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
print('Accuracy of K-NN classifier on test set:',knn.score(X_test, y_test))
print('Accuracy of K-NN classifier on train set: ',knn.score(X_train, y_train))
print("Confusion matrix:",confusion_matrix(y_test, knn_pred))
```

```
Accuracy of K-NN classifier on test set: 0.7435897435897436
Accuracy of K-NN classifier on train set:  0.7058823529411765
Confusion matrix: [[13  3]
 [ 7 16]]
```

```
print('F1 : ', f1_score(y_test, knn_pred, average=None))
print('Precision : ',precision_score(y_test, knn_pred, average=None))
print('Recall : ', recall_score(y_test, knn_pred, average=None))
```

```
F1 :  [0.72222222 0.76190476]
Precision :  [0.65       0.84210526]
Recall :  [0.8125     0.69565217]
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, knn_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, knn_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, knn_pred)))
```

```
Mean Absolute Error: 0.2564102564102564
Mean Squared Error: 0.2564102564102564
Root Mean Squared Error: 0.5063696835418333
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
#define cross-validation method to use
print(cross_val_score(knn, X, y, scoring='accuracy',cv=10).mean())
```

```
0.6673684210526315
```

5. Support Vector Machine (SVM) Classifier:

A supervised machine learning approach called Support Vector Machine (SVM) is used for both classification and regression. Finding a hyperplane in an N-dimensional space that clearly classifies the data points is the goal of the SVM method. The number of features determines the hyperplane's dimension. The hyperplane is essentially a line if there are just two input features. The hyperplane turns into a 2-D plane if there are three input features. Imagining something with more than three features gets challenging.

Kernel functions are used to efficiently identify the dot product in higher dimensions after transforming n-dimensional data to m-dimensional input, where m is significantly higher than n. Kernel's basic tenet is: In lower dimensions, a linear classifier curve transforms into a non-linear classifier curve.

a. SVM with Linear Kernel-

Linear kernel is the most fundamental kind of kernel and is often of a one-dimensional. When there are many features, it turns out to be the best choice. For text classification issues, the linear kernel is typically favored because the majority of these classification issues can be divided linearly.

The accuracy of the model is 46% and error is 0.53 which is significantly high value, which in turn means that this model is not a fit for the dataset

```
#Linear SVC
from sklearn import svm
from sklearn import metrics
classifier=svm.SVC(kernel='linear' ,gamma='auto' , C=2 )
classifier.fit(X_train, y_train)
y_predict=classifier.predict(X_test)
from sklearn.metrics import classification_report
print("Classification report:")
print(classification_report(y_test,y_predict))
print("Accuracy of Linear SVC on test set:",classifier.score(X_test,y_test))
print("Accuracy of Linear SVC on train set:",classifier.score(X_train,y_train))
print(metrics.accuracy_score(y_test,y_predict))
```

```
Classification report:
              precision    recall  f1-score   support

           1       0.38      0.50      0.43        16
           2       0.56      0.43      0.49        23

    accuracy                           0.46        39
   macro avg       0.47      0.47      0.46        39
weighted avg       0.48      0.46      0.47        39

Accuracy of Linear SVC on test set: 0.46153846153846156
Accuracy of Linear SVC on train set: 0.5686274509803921
0.46153846153846156
```

```
from sklearn import metrics
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
print('F1 : ', f1_score(y_test, y_predict, average=None))
print('Precision : ',precision_score(y_test, y_predict, average=None))
print('Recall : ', recall_score(y_test,y_predict, average=None))
```

```
F1 :  [0.43243243 0.48780488]
Precision :  [0.38095238 0.55555556]
Recall :  [0.5        0.43478261]
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_predict))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_predict))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_predict)))
```

```
Mean Absolute Error: 0.5384615384615384
Mean Squared Error: 0.5384615384615384
Root Mean Squared Error: 0.7337993857053428
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
#define cross-validation method to use
print(cross_val_score(classifier, X, y, scoring='accuracy',cv=10).mean())
```

```
0.5936842105263158
```

b. SVM with radial basis kernel

It ranks among the most popular and often used kernel functions in SVM. It is indeed frequently used with non-linear data. When there is no prior understanding of the data, it helps to make the right separation.

We can make our regression or classification line indefinitely powerful by employing the Radial Basis Kernel since it employs exponents, where e^x yields a polynomial equation with unlimited power.

```
#RBF SVC
from sklearn.svm import SVC
from sklearn.metrics import classification_report
model = SVC(kernel='rbf', random_state = 1)
model.fit(X_train, y_train)
rbfy_predict=model.predict(X_test)

print("Classification report:")
print(classification_report(y_test,rbfy_predict))
print("Accuracy of RBF SVC on test set:",model.score(X_test,y_test))
print("Accuracy of RBF SVC on train set:",model.score(X_train,y_train))
print(metrics.accuracy_score(y_test,rbfy_predict))
```

```
Classification report:
              precision    recall  f1-score   support

           1       0.50      0.50      0.50        16
           2       0.65      0.65      0.65        23

    accuracy                           0.59        39
   macro avg       0.58      0.58      0.58        39
weighted avg       0.59      0.59      0.59        39

Accuracy of RBF SVC on test set: 0.5897435897435898
Accuracy of RBF SVC on train set: 0.5947712418300654
0.5897435897435898
```

```
print('F1 : ', f1_score(y_test, rbfy_predict, average=None))
print('Precision : ',precision_score(y_test, rbfy_predict, average=None))
print('Recall : ', recall_score(y_test,rbfy_predict, average=None))
```

```
F1 :  [0.5        0.65217391]
Precision :  [0.5        0.65217391]
Recall :  [0.5        0.65217391]
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, rbfy_predict))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, rbfy_predict))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, rbfy_predict)))
```

```
Mean Absolute Error: 0.41025641025641024
Mean Squared Error: 0.41025641025641024
Root Mean Squared Error: 0.6405126152203485
```

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from numpy import mean
#define cross-validation method to use
print(cross_val_score(model, X, y, scoring='accuracy',cv=10).mean())
```

```
0.578421052631579
```

The radial kernel model's accuracy which is almost 59% is much better than the SVM with linear kernel. The mean absolute error value is 0.41 which is tolerable. But the other models seemed to have worked well with our dataset than SVM.

Briefly discuss which model fits the data better.

Based on the observations made from the estimated values we have incurred from the Cross-Validation technique, we can conclude that Random Forest Classifier has the highest accuracy and least Mean Absolute Error. Hence, it performs the best in all the cases and gives the most accurate result.

For the model that fits the data best, make predictions for at least three cases of interest.

As this is the case of medical diagnosis, accuracy cannot be the only metric to evaluate. In medical diagnosis, it is essential to predict the actual values correctly. We cannot incorrectly diagnose a patient as a regular event after the accurate diagnosis report shows that patient has autism spectrum disorder. So along with higher accuracy, we need higher recall. Comparatively, model 1 gives 82% accuracy and recall score is 0.875 on test and train data. Hence, for our specific dataset Random Forest Classifier appears to be a better match.

The second-best fit seems to be Decision tree which has recall score of 0.71 followed by Gradient boosting classifier with recall score of 0.70.

The least accuracy is given by SVM with linear kernel and radial kernel which has a recall score of 0.5.

The following code snippet compares the actual class and the predicted class using the Random Forest Classifier. As it can be seen, the predicted values are almost similar to the actual value proving that this model is indeed the best fit.

```
data = pd.DataFrame({'Actual Value' :y_test, 'Predicted Value' :rfy_pred})
data
```

	Actual Value	Predicted Value
0	1	1
1	2	2
2	1	1
3	2	1
4	1	1
5	1	1
6	2	2
7	2	2
8	1	1
9	2	2
10	2	2
11	2	2
12	2	1
13	1	1
14	1	1
15	2	2
16	1	1
17	2	2
18	1	1
19	2	2
20	1	1
21	1	1
22	2	2
23	1	1
24	1	2
25	2	2
26	1	2
27	1	1
28	2	2
29	2	2
30	1	1
31	2	1
32	2	2
33	2	2
34	2	1
35	2	2
36	2	1
37	2	2
38	2	2

4. SUMMARY AND CONCLUSION

Going back to the question that has motivated your project, how would you answer that question given the results of your analysis?

We devised a method for detecting any sort of distinguishable traits in the metabolics of ASD Patients and Controls. We expect this technology to improve healthcare delivery by automating at the expert level and extending access to medical diagnosing expertise in areas where successful diagnosis by the qualified specialists is scarce. Adding new layers to our model can enhance the metrics it, but this will introduce even more hyperparameters that need to be modified. We plan to use deep learning techniques to extend our model architecture to additional areas of medical diagnosis. The 6 models suggested in this study, which use classifiers techniques to detect autism spectrum disorder, exhibit remarkable results. Several studies on the similar dataset have recently been published. On the other hand, our models have

decent accuracy and other metrics which helps achieve state-of-the-art status. Models for efficiently classifying metabolic traits of both the cohorts and predicting the presence or absence of autism spectrum disorder were developed using various classifiers.

Think about domain experts in the field you have analyzed. What can they learn from your project? How could the results of your analysis inform their work?

People with ASD face difficulty in coping up during social interactions and with interpreting and using non-verbal and verbal communication in social contexts. Individuals with ASD may also have the difficulties like inflexible interests, insistence on sameness in environment or routine, increased or decreased reactions to sensory stimuli. Diagnosing ASD is challenging since it doesn't have any specific test that could be performed. Instead, there are several screenings and evaluations to be done over some period to identify any person affected with ASD. Identifying underlying pattern in the metabolic profiles of ASD affected persons can help make the process of diagnosing autism spectrum disorder faster. Which in turn leaves more scope for the treatment and ensure that patients get help in the earlier stages of getting affected by the disorder.

Identify one way that your project could be improved if you had more time and resources to work on this project. For example, what additional data would you gather? What alternative data cleaning decisions would you make? What additional models would you estimate?

If we had more time, we would have tried Neural Networks and made comparisons among other cohorts like Neuro-Developmental Disorders, Autistic Disorder. We would have explored the dataset more by performing statistical analysis. We would have verified if NDDs overlap more with AD or ASD, if AD and ASD overlap for certain key metabolic traits. As for the additional data, the current dataset includes the profiles of children aged under 10. Since, autism disorder can also be diagnosed in adolescent stage, we would probably have gathered metabolic profiles for adults to achieve even higher accuracy and make prediction with precision.