National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Piush Vaish
Student ID: x17122449

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Piush Vaish
Student ID:	x17122449
Programme:	Data Analytics
Year:	2019
Module:	MSc Research Project
Supervisor:	Dr. Catherine Mulwa
Submission Due Date:	16/09/2019
Project Title:	Configuration Manual
Word Count:	2094
Page Count:	28

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	7th March 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Piush Vaish
x17122449

1 Hardware

The project is implemented using the following hardware:

- Central Processing Unit (CPU) - Intel(R) Core (TM) i7-6700HQ CPU @ 2.60GHz
- RAM - 64 GB
- Cores 4
- Logical Processors - 8
- Operating System - Microsoft Windows 10

2 Data Description

To access the data, register with ADNI[1] and apply to gain access to the data. Once the access is granted, Login to ADNI and go to Download tab. Select "Study Data" followed by "Test Data" and "Data". Finally, select "Tadpole Challenge Data" to access the files. File named D1_D2.csv contains both data sets D1 (training) and D2(test). The data collected from ADNI are merged into a csv file and consists of:

- Cerebrospinal fluid (CSF) markers of amyloid-beta and tau deposition as opposed to the cerebral cortex. CSF is a clear, colorless fluid which acts a buffer in the brain. An abnormal concentration of proteins e.g., tau is an indicator of early signs of Alzheimers Disease.
- Figure 1 shows what happens to the Hippocampus[2]. Different types of radiological images including:

[1] `http://adni.loni.usc.edu/data-samples/access-data/`
[2] `https://www.helpguide.org/harvard/recognizing-and-diagnosing-alzheimers.htm`

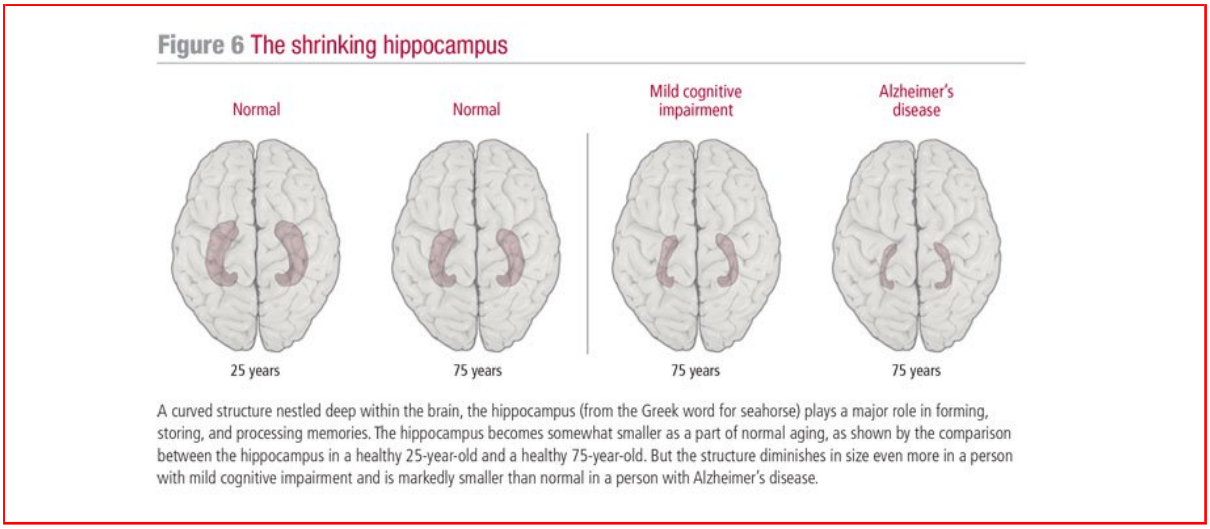


Figure 1: Shrinking Hippocampus

- Figure 2 is example of Magnetic resonance imaging (MRI)[3]. MRI measures the volume of grey and white mater of the brain. It is a good indicator of progression because it becomes abnormal. In the data set there are three types of markers of 3D sub-region measuring volumes, cortical thickness and surface areas. The markers are extracted by aligning the images with each other using a software called Freesurfer[4] and developing a cross-sectional (each subject visit is independent) or longitudinal (uses information from all the visits of a subject) pipelines.

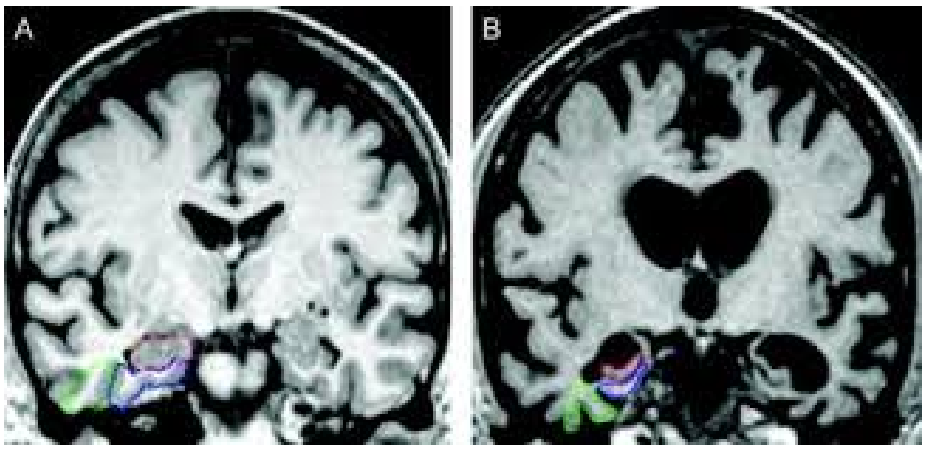


Figure 2: MRI example

- Positron emission tomography[5] (Figure 3) is generated through detection of gamma ray which are emitted by a radioactive tracer through introduction of biological active molecule. 3D images are reconstructed using a computer analysis. Scans

[3] https://www.radiologyinfo.org/en/info.cfm?pg=alzheimers
[4] https://surfer.nmr.mgh.harvard.edu/
[5] https://www.neurologyadvisor.com/topics/neurodegenerative-diseases/pet-scans-distinguish-alzheimer-disease-from-other-neurodegenerative-diseases/

are of different types depending on cellular and molecular processes. The molecular processes are the first which become abnormal and hence useful from healthy control to progress to MCI or not. However, these have lower spatial resolution than MRI. The images in the data set are have their frames processed, averaged across the dynamic range and standardized. Standardised uptake value ratio (SUVR) measures for relevant regions-of- interest are extracted after registering the PET images to corresponding MRI using the SPM5 software.

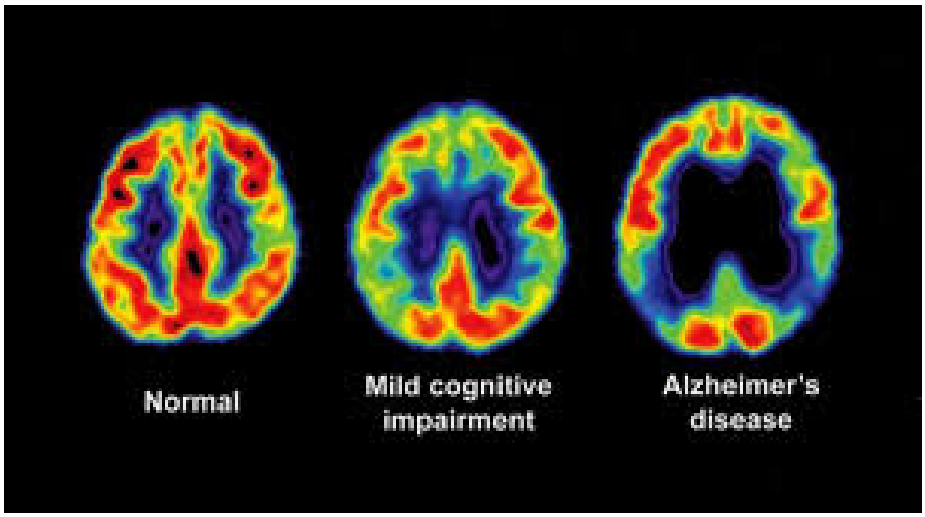
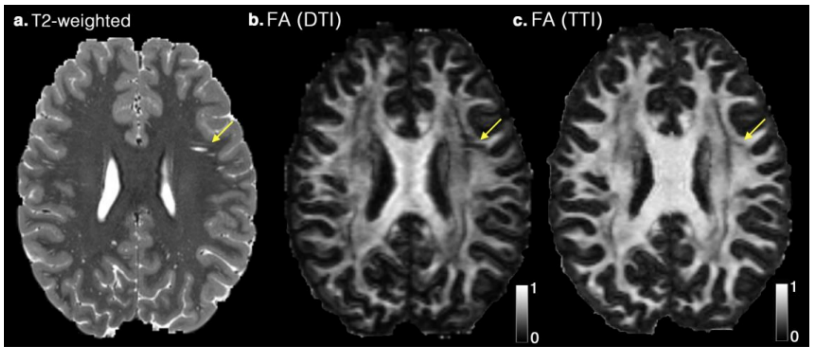


Figure 3: PET example

- Diffusion tensor imaging (Figure 4) measures the degeneration of white matter [6]. The scans are corrected for head motion and eddy-current distortion, skull-stripped, EPI- corrected, and finally aligned to the T1 scans.



Supplemental Figure 3. An example of the influence of PVS on fractional anisotropy (FA) map. **(a)** T2-weighted image, **(b)** Voxel FA, derived from diffusion tensor imaging (DTI), and **(c)** Tissue FA, derived from tissue tensor imaging (TTI). An example with high amount of PVS is highlighted with yellow arrows. Note that FA values from TTI are higher throughout the brain.

Figure 4: DTI example

[6] http://blogs.discovermagazine.com/neuroskeptic/2018/09/01/white-matter-worries-dti/

- Cognitive tests such as ADAS11, the Mini-Mental State Examination (MMSE) acquired in the presence of a clinical expert. Cognitive tests measure decline in a direct and quantifiable manner.
- Genetic information such as alipoprotein E4 (APOE4) status.
- General demographic information such as age, gender and education. Age is an important factor. Females are also more likely to develop the disease than men. Other factors such as smoking, diabetes depression, head injuries also increase the risk of developing the disease.

3 Installation of Anaconda

The project uses the open-source Anaconda Distribution[7].

Figure 5: Anaconda

3.1 Installing on Windows

- Download the Anaconda installer[8]
- Double click the installer to launch.
- Click Next button.
- Read the licensing terms and click "I Agree".
- Select "Just Me" and click Next button.
- Select a destination folder to install Anaconda and click Next button.

[7] https://docs.anaconda.com/anaconda/
[8] https://www.anaconda.com/download/

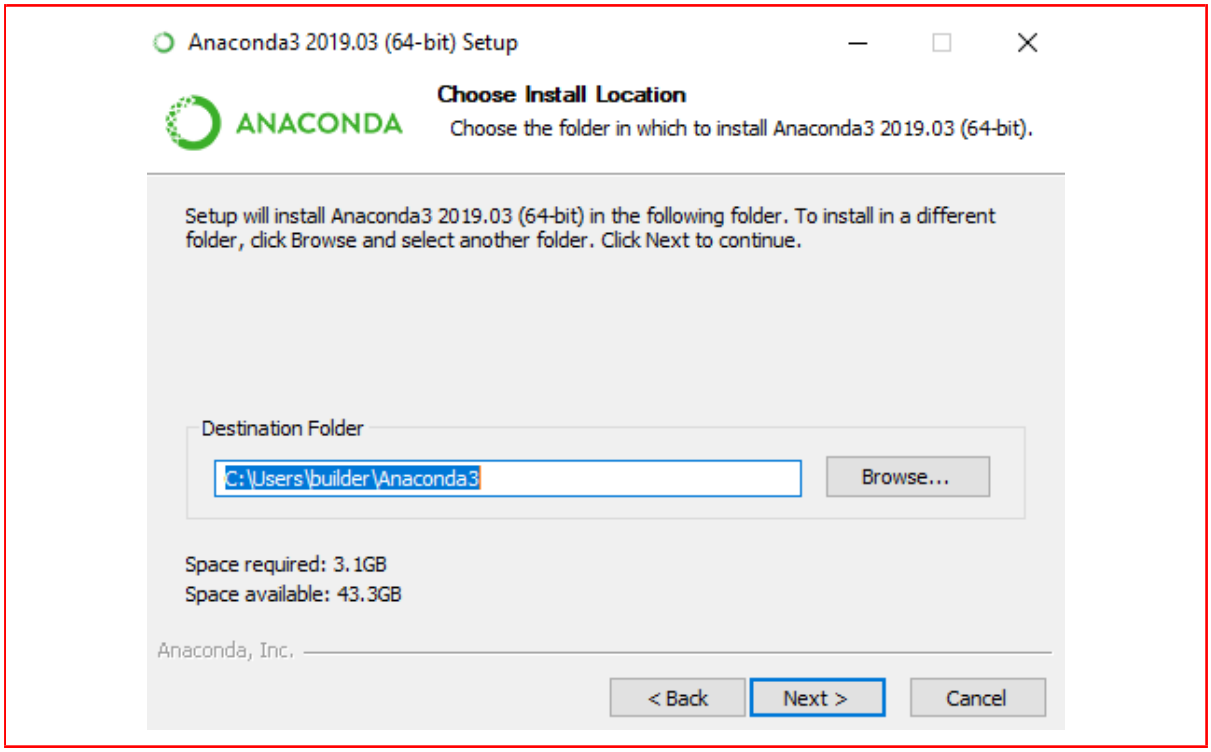


Figure 6: Anaconda Installation

- Choose to add Anaconda to the PATH.
- Use software through Anaconda Prompt from the Start Menu.

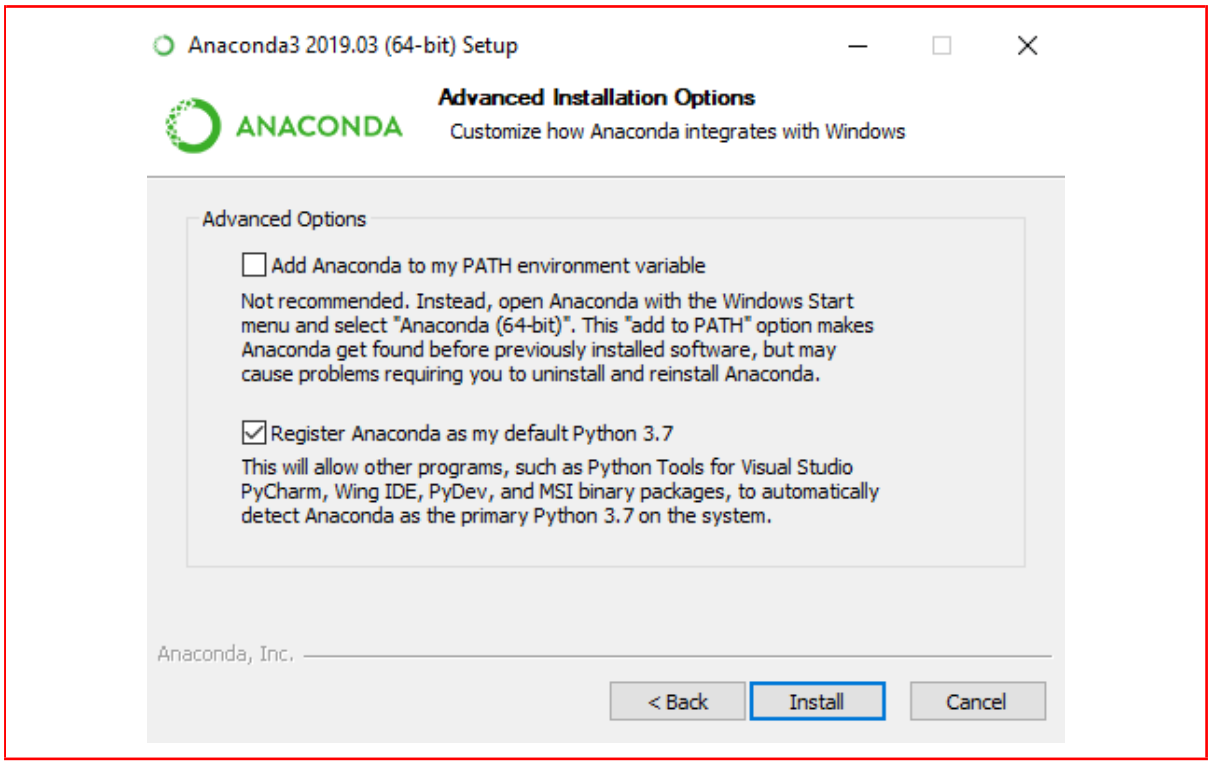


Figure 7: Anaconda Installation (Advanced)

- Choose whether to register Anaconda as your default Python.
- Click the Install button.
- Click the Next button.
- After a successful installation, Figure 8 is shown.

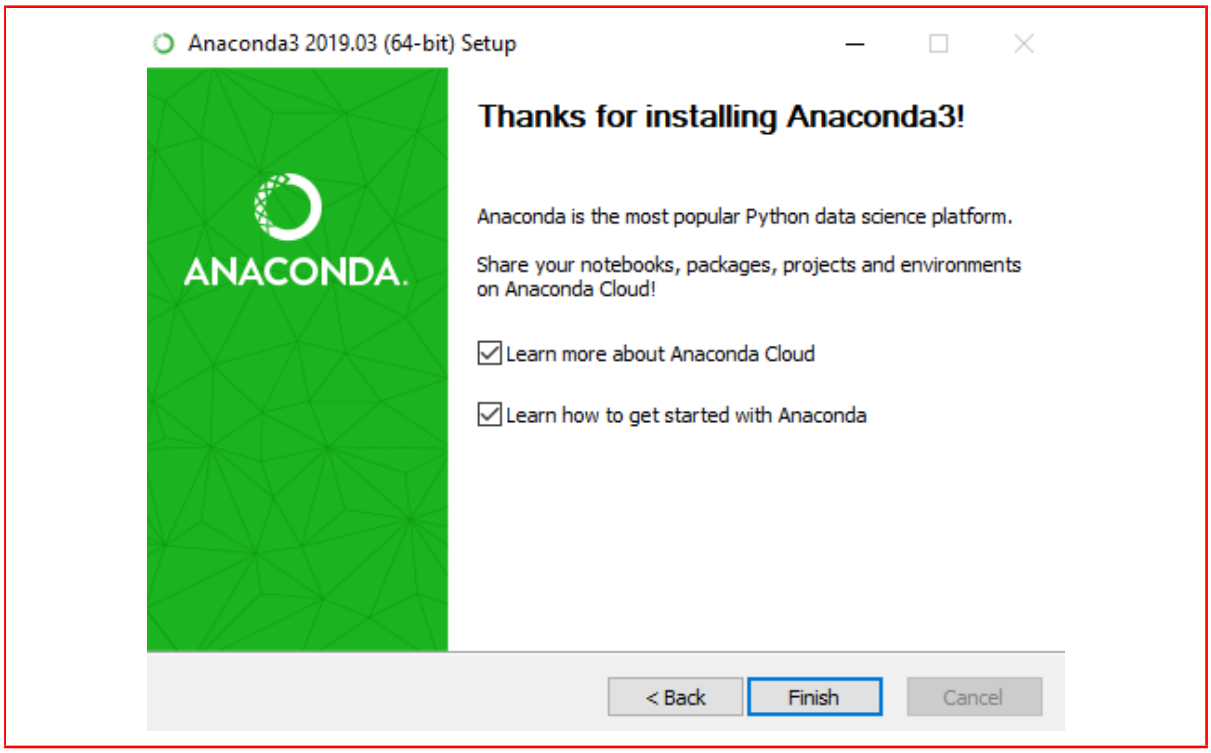


Figure 8: Anaconda Installation (Success)

4 Setting up GitHub Desktop

Before setting up GitHub Desktop, ensure there is already a GitHub or GitHub Enterprise account[9].

- Visit the GitHub Desktop download page.

Figure 9: GitHub

[9] https://help.github.com/en/articles/signing-up-for-a-new-github-account/

- Choose Download for Windows [10].
- Double-click GitHub Desktop where the file has been downloaded.
- Click Install.

5 Cloning the GitHub Repository

The code for the project is available at GitHub [11]

- Navigate to the repository as shown in Figure 10.

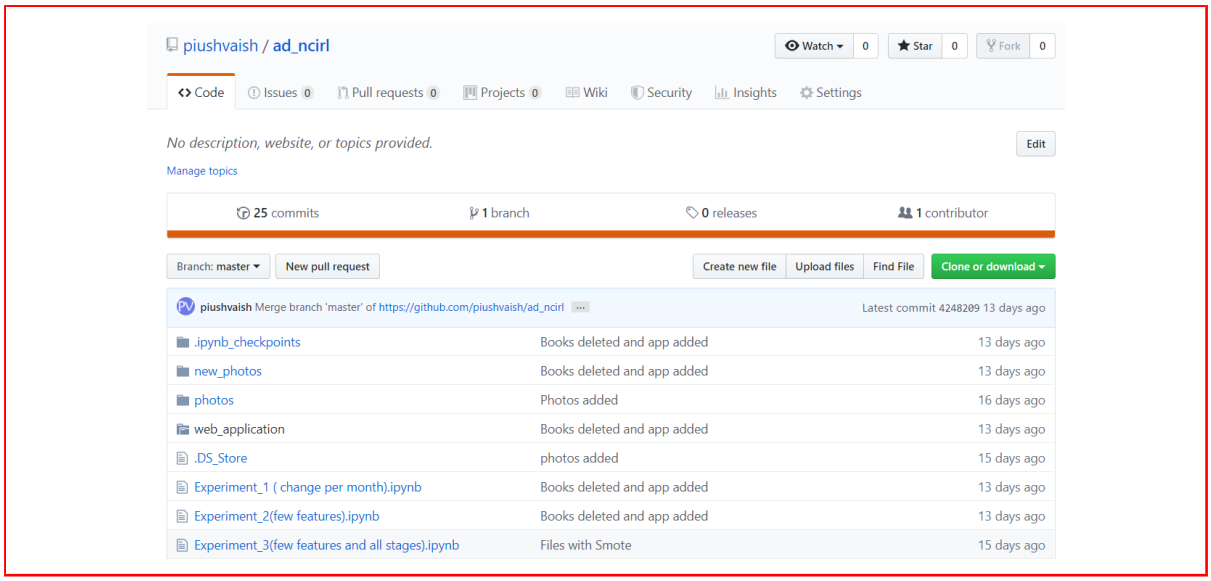


Figure 10: Cloning GitHub Repository

- click Clone or download button.
- Click to copy the clone URL as shown in Figure 11.

[10] `https://desktop.github.com/`
[11] `https://github.com/piushvaish/ad_ncirl`

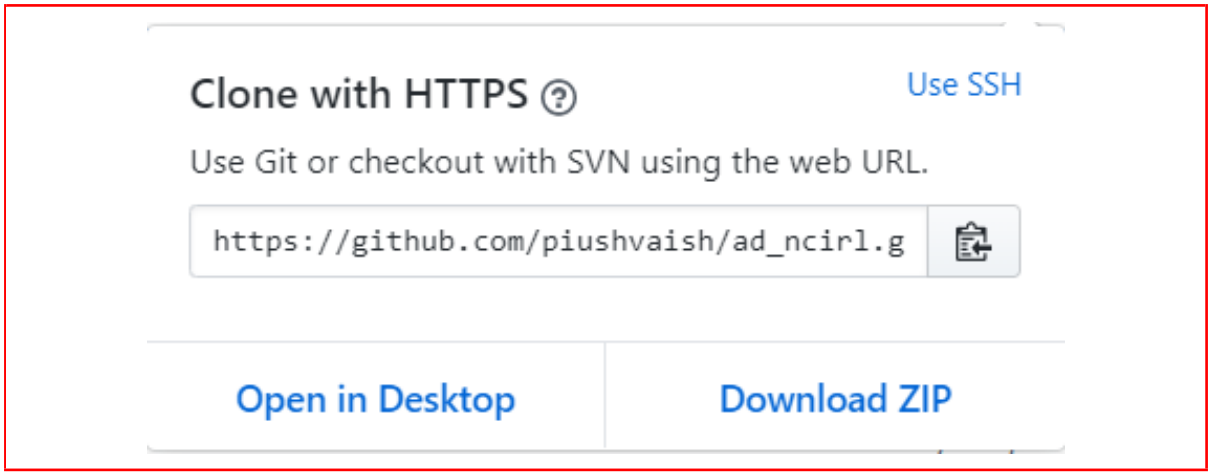


Figure 11: Cloning GiHub Repository

- Open Git Bash.
- Change to where the cloned directory is to be made.
- Type git clone, and then paste the URL from section 5.
- Press Enter.

A local clone will be created.

6 Create a Virtual Environment

A key reason for creating an isolated project environment is to facilitate the installation of new packages without interfering with the system's Python installation or any other project installation.

- In the terminal client enter the following
  conda create n ad_ncirl python=3.6 anaconda
- Press y to install the Python version and all the associated anaconda packaged libraries at
  path__to_your_anaconda_location/anaconda/envs/ad_ncirl

6.1 Activate Virtual Environment

To activate virtual environment, type
source activate ad_ncirl

6.2 Install Additional Python Packages to a Virtual Environment

To install additional packages , enter:
conda install -n [package]

- alabaster==0.7.10
- anaconda-client==1.6.14
- Babel==2.5.3
- backcall==0.1.0
- bleach==2.1.3
- certifi==2018.1.18
- clyent==1.2.2
- colorama==0.3.9
- cycler==0.10.0
- decorator==4.3.0
- docutils==0.14
- entrypoints==0.2.3
- future==0.16.0
- html5lib==1.0.1
- imageio==2.5.0
- imagesize==1.0.0
- imbalanced-learn==0.5.0
- imblearn==0.0
- ipykernel==4.8.2
- ipython==6.3.1
- ipython-genutils==0.2.0
- ipywidgets==7.2.1
- jedi==0.12.0
- Jinja2==2.10
- joblib==0.13.2
- jsonschema==2.6.0
- jupyter==1.0.0
- jupyter-client==5.2.3
- jupyter-console==5.2.0

- jupyter-core==4.4.0
- MarkupSafe==1.0
- matplotlib==2.1.1
- mistune==0.8.3
- mkl-fft==1.0.0
- mkl-random==1.0.1
- nb-anacondacloud==1.4.0
- nb-conda==2.2.0
- nb-conda-kernels==2.1.0
- nbconvert==5.3.1
- nbformat==4.4.0
- nbpresent==3.0.2
- networkx==2.3
- nltk==3.2.5
- notebook==5.4.1
- numpy==1.16.4
- numpydoc==0.8.0
- packaging==17.1
- pandas==0.22.0
- pandocfilters==1.4.2
- parso==0.2.0
- pickleshare==0.7.4
- Pillow==6.0.0
- plotly==3.10.0
- pomegranate==0.9.0
- prompt-toolkit==1.0.15
- pyasn1==0.3.7
- pydot==1.2.4
- pydotplus==2.0.2

- Pygments==2.2.0
- pyparsing==2.2.0
- python-dateutil==2.6.1
- pytz==2018.4
- PyWavelets==1.0.3
- pywinpty==0.5.1
- PyYAML==3.12
- pyzmq==17.0.0
- qtconsole==4.3.1
- requests==2.14.2
- requests-toolbelt==0.8.0
- retrying==1.3.3
- scikit-image==0.15.0
- scikit-learn==0.21.2
- scipy==1.3.0
- seaborn==0.8.1
- Send2Trash==1.5.0
- shap==0.29.3
- simplegeneric==0.8.1
- six==1.11.0
- snowballstemmer==1.2.1
- Sphinx==1.7.2
- sphinxcontrib-websupport==1.0.1
- terminado==0.8.1
- testpath==0.3.1
- tornado==5.0.1
- tqdm==4.32.2
- traitlets==4.3.2
- typing==3.6.4

- udacity-pa==0.2.8
- wcwidth==0.1.7
- webencodings==0.5.1
- widgetsnbextension==3.2.1
- win-unicode-console==0.5
- wincertstore==0.2
- xgboost==0.90

6.3 To Start a New Jupyter Notebook

- Go to the Windows start menu and select Anaconda Prompt under Anaconda3.
- At the Anaconda Prompt type:
  jupyter notebook
- This will start a Jupyter notebook. The output in the terminal will look something like below:

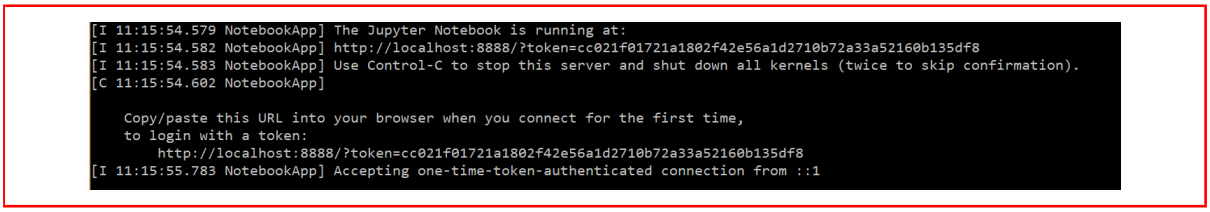


Figure 12: Jupyter Notebook

- A web browser should open and you should be able to see the Jupyter file browser

7 Running Jupyter Notebooks

The web browser go to the folder where the project is cloned. Open a Jupyter Notebook and click to run the kernel.

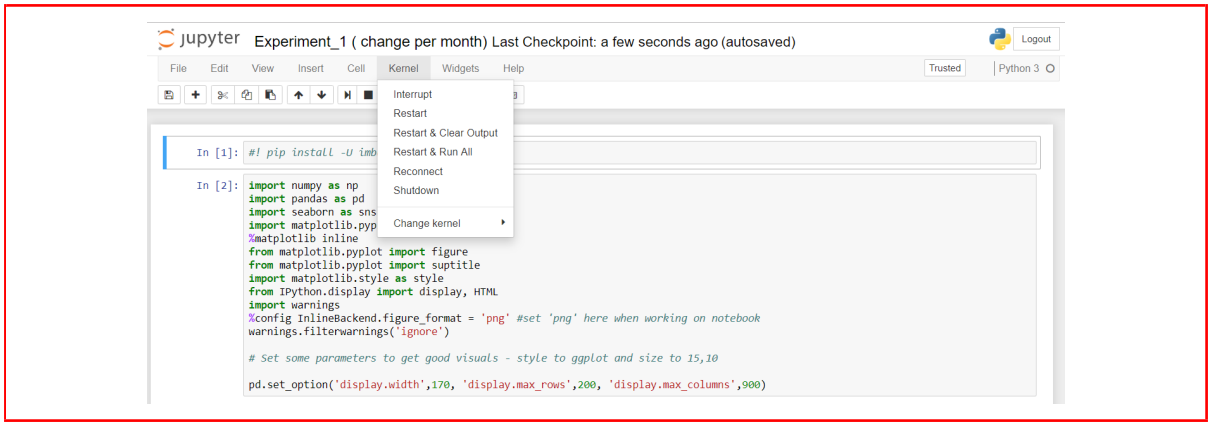


Figure 13: Run Jupyter Notebook

8 Code Examples

Figure 14 shows code for importing general python libraries.

Figure 14: Import Python Libraries

Figure 15 shows code for checking columns with missing values and their data type.

```
import re
missing_values = []
nonumeric_values = []

print ("Data INFORMATION")
print ("=======================\n")

for column in df1:
    # Find all the unique feature values
    uniq = df1[column].unique()
    print ("'{}' has {} unique values" .format(column,uniq.size))
    if (uniq.size > 10):
        print("~~Listing up to 10 unique values~~")
    print (uniq[0:10])
    print ("\n---------------------------------------------------------------------\n")

    # Find features with missing values
    if (True in pd.isnull(uniq)):
        s = "{} has {} missing" .format(column, pd.isnull(df1[column]).sum())
        missing_values.append(s)

    # Find features with non-numeric values
    for i in range (1, np.prod(uniq.shape)):
        if (re.match('nan', str(uniq[i]))):
            break
        if not (re.search('(^\d+\.?\d*$)|(^\d*\.?\d+$)', str(uniq[i]))):
            nonumeric_values.append(column)
            break

print ("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n")
print ("Features with missing values:\n{}\n\n" .format(missing_values))
print ("Features with non-numeric values:\n{}" .format(nonumeric_values))
print ("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n")
```

Figure 15: Check Columns with Missing and Their Data Type

Figure 16 shows code for replacing values making only three categories.

```
df2 = df2.replace({'NL to MCI': 'MCI', 'MCI to Dementia': 'Dementia', 'MCI to NL' : 'NL',
                   'NL to Dementia': 'Dementia', 'Dementia to MCI': 'MCI'})
```

Figure 16: Replace Values

Figure 17 shows code for creating dummy variables for categorical features.

```
categorial_cols = [
    'PTGENDER', 'PTETHCAT', 'PTRACCAT', 'PTMARRY']

for cc in categorial_cols:
    dummies = pd.get_dummies(df3[cc])
    dummies = dummies.add_prefix("{}#".format(cc))
    df3.drop(cc, axis=1, inplace=True)
    df3 = df3.join(dummies)
```

Figure 17: Dummy Variables

Figure 18 shows code for importing scikit-learn libraries.

```
from sklearn import model_selection
from sklearn import mixture
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from itertools import product
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.multiclass import OneVsRestClassifier
```

Figure 18: Import Scikit-learn Libraries

Figure 18 shows code for creating features for monthly change.

```
df2['FDG_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['FDG'] - df2['FDG'].shift()) /df2['Month'] , 0),2)
df2['AV45_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['AV45'] - df2['AV45'].shift()) /df2['Month'] , 0),2)
df2['CDRSB_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['CDRSB'] - df2['CDRSB'].shift()) /df2['Month'] , 0),2)
df2['ADAS11_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['ADAS11'] - df2['ADAS11'].shift()) /df2['Month'] , 0),2
)
df2['MMSE_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['MMSE'] - df2['MMSE'].shift()) /df2['Month'] , 0),2)
df2['RAVLT_immediate_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['RAVLT_immediate'] - df2['RAVLT_immediate'].sh
ift()) /df2['Month'] , 0),2)
df2['Hippocampus_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['Hippocampus'] - df2['Hippocampus'].shift()) /df2[
'Month'] , 0),2)
df2['WholeBrain_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['WholeBrain'] - df2['WholeBrain'].shift()) /df2['Mo
nth'] , 0),2)
df2['Entorhinal_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['Entorhinal'] - df2['Entorhinal'].shift()) /df2['Mo
nth'] , 0),2)
df2['MidTemp_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['MidTemp'] - df2['MidTemp'].shift()) /df2['Month'] , 0
),2)
df2['ABETA_UPENNBIOMK9_04_19_17_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['ABETA_UPENNBIOMK9_04_19_17'] - df2
['ABETA_UPENNBIOMK9_04_19_17'].shift()) /df2['Month'] , 0),2)
df2['TAU_UPENNBIOMK9_04_19_17_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['TAU_UPENNBIOMK9_04_19_17'] - df2['TA
U_UPENNBIOMK9_04_19_17'].shift()) /df2['Month'] , 0),2)
df2['PTAU_UPENNBIOMK9_04_19_17_CHANGE'] = np.round(np.where(df2.RID == df2.RID.shift(),(df2['PTAU_UPENNBIOMK9_04_19_17'] - df2[
'PTAU_UPENNBIOMK9_04_19_17'].shift()) /df2['Month'] , 0),2)
```

Figure 19: Code for creating features for monthly change

Figure 20 shows code for running the different machine learning algorithms and then generating a confusion matrix.

```
from sklearn.metrics import confusion_matrix
# prepare models
models = []
models.append(('LR', OneVsRestClassifier(LogisticRegression())))
models.append(('LDA', OneVsRestClassifier(LinearDiscriminantAnalysis())))
models.append(('KNN', OneVsRestClassifier(KNeighborsClassifier())))
models.append(('CART', OneVsRestClassifier(DecisionTreeClassifier())))
models.append(('RF',OneVsRestClassifier(RandomForestClassifier(max_depth=1))))
models.append(('NB', OneVsRestClassifier(GaussianNB())))
models.append(('Neural Network', OneVsRestClassifier(MLPClassifier())))
models.append(('SVM', OneVsRestClassifier(SVC())))
# evaluate each model in turn
results = []
names = []
cm =[]
y_actu_arr = (Y_test)
for name , model in models:
    model.fit(X_train, Y_train)
    y_pred = model.predict(X_test)

    results.append(confusion_matrix(y_actu_arr,y_pred))
    names.append(name)
print()
print("Variable :{}".format(i))


for j, k in zip(names, results):
    print("{} Confusion Matrix : {}".format(j,k))
```

Figure 20: Run Different Machine Learning Algorithms and Generate Confusion Matrix

When you try to use roc_auc_score on a multiclass classification, you will receive the following error:

```
roc_auc_score(y_test,y_pred)

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-5ce299873d02> in <module>()
----> 1 roc_auc_score(y_test,y_pred)

~\Anaconda3\envs\dsi\lib\site-packages\sklearn\metrics\ranking.py in roc_auc_score(y_true, y_score, average, sample_weight)
    275     return _average_binary_score(
    276         _binary_roc_auc_score, y_true, y_score, average,
--> 277         sample_weight=sample_weight)
    278
    279

~\Anaconda3\envs\dsi\lib\site-packages\sklearn\metrics\base.py in _average_binary_score(binary_metric, y_true, y_score, averag
e, sample_weight)
     70     y_type = type_of_target(y_true)
     71     if y_type not in ("binary", "multilabel-indicator"):
---> 72         raise ValueError("{0} format is not supported".format(y_type))
     73
     74     if y_type == "binary":

ValueError: multiclass format is not supported
```

Figure 21: AUROC Error

Therefore, the code for measuring the performance of multiclass classification as follows:

Averaged Multiclass ROC AUC Score

In [61]:
```
#https://medium.com/@plog397/auc-roc-curve-scoring-function-for-multi-class-classification-9822871a6659
from sklearn import preprocessing
def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = preprocessing.LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)
    return roc_auc_score(y_test, y_pred, average=average)
```

Figure 22: Average AUROC Score

Multiclass ROC Dictionary

```
from sklearn.metrics import roc_auc_score
def multiclass_roc_dict(model):
    selected_classifier = model
    selected_classifier.fit(X_train, Y_train)
    y_pred = model.predict(X_test)
    #creating a set of all the unique classes using the actual class list
    unique_class = set(test['DX'].values)
    roc_auc_dict = {}
    for per_class in unique_class:
        #creating a list of all the classes except the current class
        other_class = [x for x in unique_class if x != per_class]

        #marking the current class as 1 and all other classes as 0
        new_actual_class = [0 if x in other_class else 1 for x in Y_test]
        new_pred_class = [0 if x in other_class else 1 for x in y_pred]

        #using the sklearn metrics method to calculate the roc_auc_score
        roc_auc = roc_auc_score(new_actual_class, new_pred_class, average = 'macro')
        roc_auc_dict[per_class] = roc_auc
    return roc_auc_dict
```

Figure 23: Dictionary of Multiclass AUROC Score

The code for plotting confusion matrix and AUROC curve involved factoring of the code to ensure it displays well. However, it is too many lines to fit and is not eligible if the zoom is decreased. Please visit the GitHub page[12].

Figure 24 is the code for grid search for XGBoost.

[12] `https://github.com/piushvaish/ad_ncirl`

```
param_xgb = {
        'silent': [False],
        'max_depth': [6, 10, 15, 20],
        'learning_rate': [0.001, 0.01, 0.1, 0.2, 0,3],
        'subsample': [0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'colsample_bytree': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'colsample_bylevel': [0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
        'min_child_weight': [0.5, 1.0, 3.0, 5.0, 7.0, 10.0],
        'gamma': [0, 0.25, 0.5, 1.0],
        'reg_lambda': [0.1, 1.0, 5.0, 10.0, 50.0, 100.0],
        'n_estimators': [50,100,120]}
```

```
from sklearn.model_selection import GridSearchCV

model_xgb = xgb.XGBClassifier()
xgb_random = RandomizedSearchCV(estimator = model_xgb, param_distributions = param_xgb, n_iter = 100, cv = FOLDS,
                                verbose=2, random_state=42, n_jobs = -1, scoring='accuracy')
xgb_random.fit(train_X_scaled, train_y)
```

```
xgb_random.best_params_
```

Figure 24: Grid Search for XGBoost

Figure 25 is the code for grid search for Gradient Boosting Classifier.

```
parametros_gb = {
    "loss":["deviance"],
    "learning_rate": [0.01, 0.025, 0.005,0.5, 0.075, 0.1, 0.15, 0.2,0.3,0.8,0.9],
    "min_samples_split": [0.01, 0.025, 0.005,0.4,0.5, 0.075, 0.1, 0.15, 0.2,0.3,0.8,0.9],
    "min_samples_leaf": [1,2,3,5,8,10,15,20,40,50,55,60,65,70,80,85,90,100],
    "max_depth":[3,5,8,10,15,20,25,30,40,50],
    "max_features":["log2","sqrt"],
    "criterion": ["friedman_mse",  "mae"],
    "subsample":[0.5, 0.618, 0.8, 0.85, 0.9, 0.95, 1.0],
    "n_estimators":range(1,100)
    }

model_gb= GradientBoostingClassifier()


gb_random = RandomizedSearchCV(estimator = model_gb, param_distributions = parametros_gb, n_iter = 100, cv = FOLDS,
                                verbose=2, random_state=42, n_jobs = -1, scoring='accuracy')
gb_random.fit(train_X_scaled, train_y)
```

Figure 25: Grid Search for Gradient Boosting Classifier

Figure 26 is the code for feature selection using variance threshold.

```
#Find all features with more than 90% variance in values.
threshold = 0.90
vt = VarianceThreshold().fit(train_X )

# Find feature names
feat_var_threshold = train_X.columns[vt.variances_ > threshold * (1-threshold)]
feat_var_threshold [:20]
```

Figure 26: Feature Selection using Variance Threshold

Figure 27 is the code for ensemble of classifiers with tuned hyper parameters.

```
from sklearn.ensemble import BaggingClassifier, ExtraTreesClassifier, GradientBoostingClassifier,
VotingClassifier, RandomForestClassifier, AdaBoostClassifier
import xgboost as xgb
#Voting Ensemble
# Create sub models
estimators = []

estimators.append(('et', ExtraTreesClassifier(n_estimators=60,min_samples_split=8,max_features='sqrt',max_depth= 37)))
estimators.append(('gbm', GradientBoostingClassifier(subsample = 0.95,n_estimators= 32,
                  min_samples_split = 0.01,
                  min_samples_leaf = 10,
                  max_features = 'sqrt',
                  max_depth = 25,
                  loss = 'deviance',
                  learning_rate = 0.025,
                  criterion= 'friedman_mse')))
estimators.append(('rf', RandomForestClassifier(n_estimators=49,min_samples_split=25,max_features='auto',max_depth= 22)))
estimators.append(('ada', AdaBoostClassifier(n_estimators=37,learning_rate=0.2)))
#estimators.append(('svm', SVC(C = 25, gamma= 0.01, kernel ='linear')))
estimators.append(('xgb', xgb.XGBClassifier(psubsample= 0.6,
 silent= False,
 reg_lambda =0.1,
 n_estimators= 120,
 min_child_weight= 1.0,
 max_depth = 6,
 learning_rate= 0.01,
 gamma= 0.25,
 colsample_bytree=0.9,
 colsample_bylevel= 0.4)))

# create the ensemble model
ensemble = VotingClassifier(estimators, voting='soft', weights=[2,3,3,1,3])
```

Figure 27: Ensemble of Classifiers with Tuned Hyperparameters

Figure 18 shows code for creating decision tree.

```
predictors=list(train_X.columns)
from sklearn.externals.six import StringIO
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(min_samples_leaf=0.125, min_samples_split=0.125)
dtc = dtc.fit(train_X_scaled, train_y)
from sklearn import tree
tree.export_graphviz(dtc, out_file="tree_few_features.dot", feature_names=predictors, proportion=True)
```

Figure 28: Code for Creating Decision Tree

Figure 29 shows code for generating classification report and an example for ensemble of classifiers.

```
print("Ensemble Results with Parameter Tuning")
print(multiclass_roc_dict(test_y, preds_y))
```

```
Ensemble Results with Parameter Tuning
{1: 0.7241473547662742, 2: 0.5966558007025501, 3: 0.8904586131301045}
```

```
from sklearn.metrics import classification_report

print("Classification Report : {}".format(classification_report(test_y, preds_y, labels=[1, 2, 3])))
```

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

           1       0.94      0.47      0.62      2059
           2       0.58      0.60      0.59      2481
           3       0.40      0.99      0.57       637

    accuracy                           0.59      5177
   macro avg       0.64      0.69      0.59      5177
weighted avg       0.70      0.59      0.60      5177
```

Figure 29: Code for Generating Classification Report

Figure 30, Figure 31, Figure 32 and Figure 33 is the code for generating the SHAP values for explaining the output of the model.

```
import shap
import xgboost
# load JS visualization code to notebook
shap.initjs()
model = xgboost.train({"learning_rate": 0.01, "n_estimators": 100, "eval_set":[(test_X_scaled, test_y)] },
                      xgboost.DMatrix(train_X_scaled, train_y), 100)
# explain the model's predictions using SHAP values
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(train_X)

# visualize the first prediction's explanation (use matplotlib=True to avoid Javascript)
shap.force_plot(explainer.expected_value, shap_values[0,:], train_X.iloc[0,:],matplotlib=True)
```

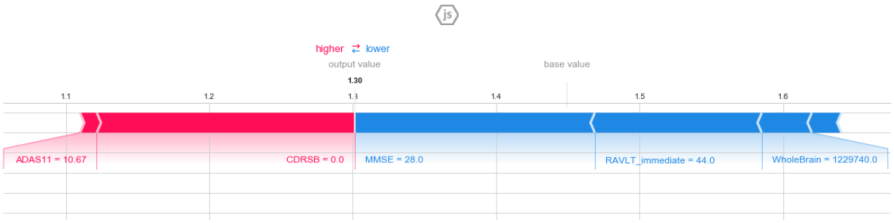


Figure 30: SHAP Values (1)

```
# visualize the training set predictions
shap.force_plot(explainer.expected_value, shap_values, train_X)
```

Figure 31: SHAP Values (2)

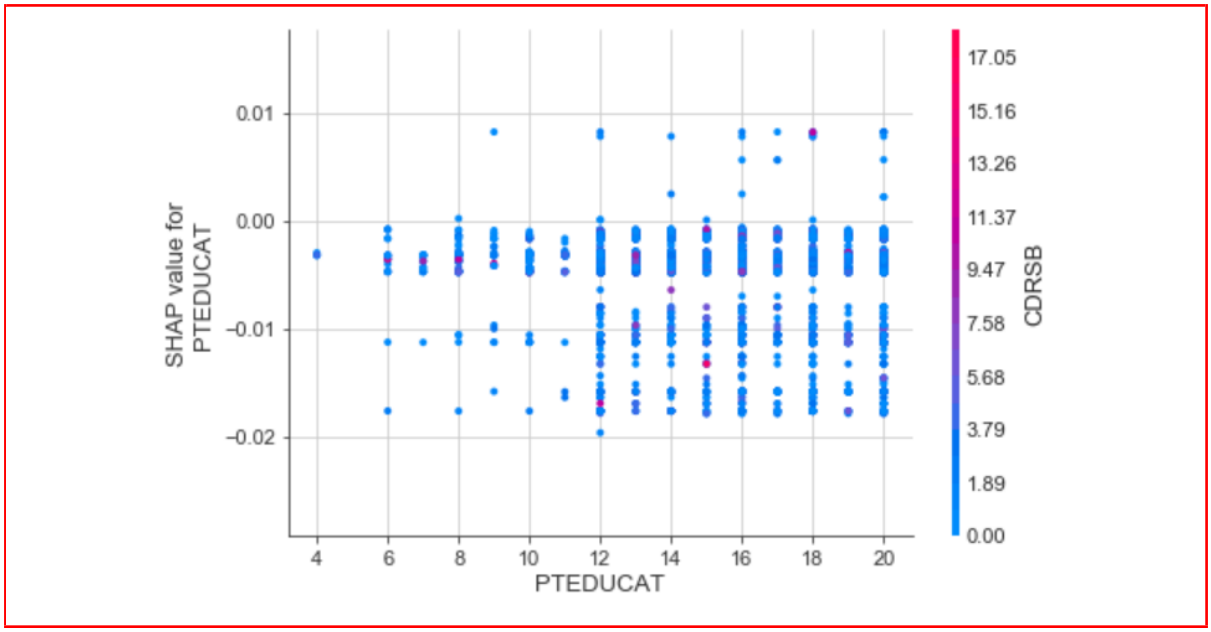


Figure 32: SHAP Values (3)

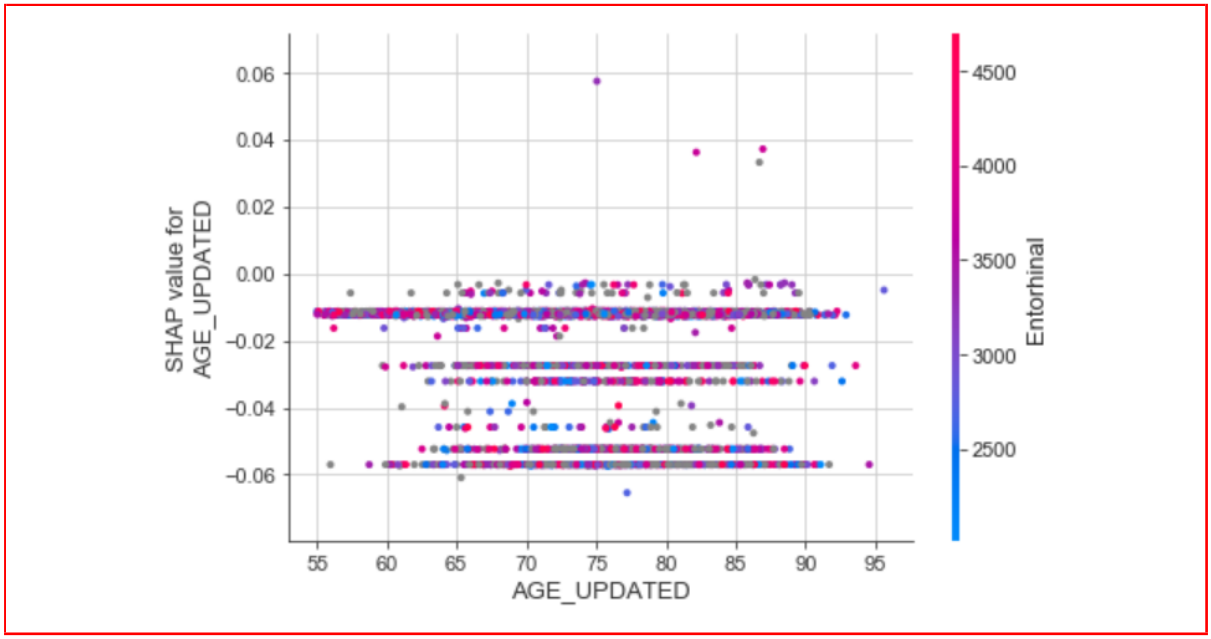


Figure 33: SHAP Values (4)

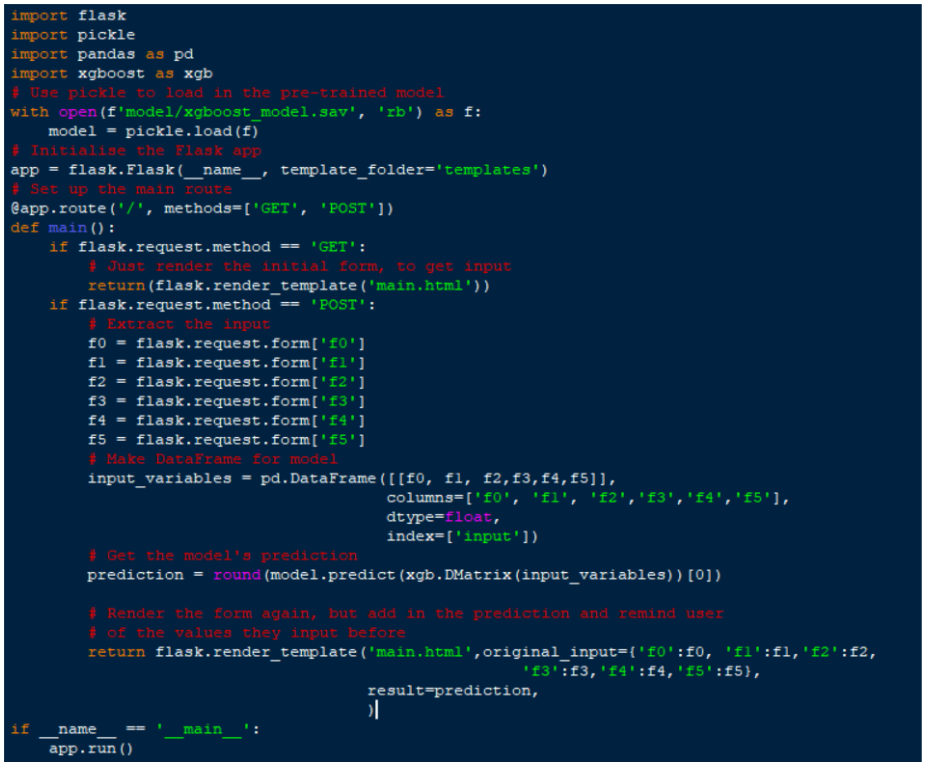
9 Implementation of Web-based Application

Web-based application is developed using Flask microframework and deployed using the Heroku platform. Flask[13] is a framework which when loaded in automatically executes the routine code. Application runs on the Heroku[14] server, send out web pages and

[13] `http://flask.pocoo.org/`
[14] `https://www.heroku.com/`

process input from users.

The Python code (Figure 34) load the model, get user input from a web form, do predictions, return results. It also assigns the pickled model to a variable and import the required packages.



```
import flask
import pickle
import pandas as pd
import xgboost as xgb
# Use pickle to load in the pre-trained model
with open(f'model/xgboost_model.sav', 'rb') as f:
    model = pickle.load(f)
# Initialise the Flask app
app = flask.Flask(__name__, template_folder='templates')
# Set up the main route
@app.route('/', methods=['GET', 'POST'])
def main():
    if flask.request.method == 'GET':
        # Just render the initial form, to get input
        return(flask.render_template('main.html'))
    if flask.request.method == 'POST':
        # Extract the input
        f0 = flask.request.form['f0']
        f1 = flask.request.form['f1']
        f2 = flask.request.form['f2']
        f3 = flask.request.form['f3']
        f4 = flask.request.form['f4']
        f5 = flask.request.form['f5']
        # Make DataFrame for model
        input_variables = pd.DataFrame([[f0, f1, f2,f3,f4,f5]],
                                       columns=['f0', 'f1', 'f2','f3','f4','f5'],
                                       dtype=float,
                                       index=['input'])
        # Get the model's prediction
        prediction = round(model.predict(xgb.DMatrix(input_variables))[0])

        # Render the form again, but add in the prediction and remind user
        # of the values they input before
        return flask.render_template('main.html',original_input={'f0':f0, 'f1':f1,'f2':f2,
                                                                  'f3':f3,'f4':f4,'f5':f5},
                                     result=prediction,
                                     )
if __name__ == '__main__':
    app.run()
```

Figure 34: app.py

(Figure 35) is an HTML templates that Flask renders and allow the user to input their own data and present the results.

The user fills in the form on the page and clicks on a submit button, Flask receives a request, extracts the input, runs it through the model and finally render (Figure 35) with the results in place. The user can then view that result and submit more data for processing. The important components are the form (The action attribute that tells Flask which route (and therefore function) should be called when the form is submitted) and some input checks.

```
<!doctype html>
<html>
<head>
    <title>Prediciting Stages of Alzhemier's Disease</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}">
</head>
    </header>
    <div class="container">

<h3> Web Application for Alzheimer's Disease.</h3>

</div>
 <div class="container">
<form action="{{ url_for('main') }}" method="POST">
    <fieldset>
        <legend>Input values:</legend>
        CDRSB:
        <input name="f0" type="number" step="0.5" min="0" max='18' oninput="this.value = Math.abs(this.value)" required>
        <br>
        <br> ADAS11:
        <input name="f1" type="number" step="0.33" min="0" max='70' oninput="this.value = Math.abs(this.value)" required>
        <br>
        <br> MMSE:
        <input name="f2" type="number" step="1" min="0" max='30' oninput="this.value = Math.abs(this.value)" required>
        <br>
        <br> RAVLT_immediate:
        <input name="f3" type="number" step="1" min="0" max='75' oninput="this.value = Math.abs(this.value)" required>
        <br>
        <br> Whole Brain:
        <input name="f4" type="number" step="100" min="649091" max='1486040' oninput="this.value = Math.abs(this.value)" required>
        <br>
        <br> AGE_UPDATED:
        <input name="f5" type="number" step="0.1" min="55" max='75' oninput="this.value = Math.abs(this.value)" required>
        <br>
        <br>
        <input type="submit">
    </fieldset>
</form>
<br>
```

Figure 35: main.HTML

Figure 36 is a CSS file for appearance.

```css
body {
  margin: 0;
  padding: 0;
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #444;
}
/*
 * Formatting the header area
 */
header {
  background-color: #DFB887;
  height: 35px;
  width: 100%;
  opacity: .9;
  margin-bottom: 10px;
}
header h1.logo {
  margin: 0;
  font-size: 1.7em;
  color: #fff;
  text-transform: uppercase;
  float: left;
}
header h1.logo:hover {
  color: #fff;
  text-decoration: none;
}
/*
 * Centering the body content
 */
.container {
  width: 1200px;
  margin: 0 auto;
}
div.home {
  padding: 10px 0 30px 0;
  background-color: #E6E6FA;
  -webkit-border-radius: 6px;
     -moz-border-radius: 6px;
          border-radius: 6px;
}
div.about {
  padding: 10px 0 30px 0;
  background-color: #E6E6FA;
  -webkit-border-radius: 6px;
     -moz-border-radius: 6px;
          border-radius: 6px;
}
```

Figure 36: main.CSS

Figure 37 shows the classification report for few models.

```
LR Classification Report :              precision    recall  f1-score   support

           1       0.49      0.65      0.56      2059
           2       0.60      0.15      0.25      2481
           3       0.22      0.62      0.32       637

   micro avg       0.41      0.41      0.41      5177
   macro avg       0.44      0.48      0.38      5177
weighted avg       0.51      0.41      0.38      5177

LDA Classification Report :              precision    recall  f1-score   support

           1       0.49      0.65      0.56      2059
           2       0.59      0.15      0.24      2481
           3       0.22      0.62      0.32       637

   micro avg       0.41      0.41      0.41      5177
   macro avg       0.43      0.47      0.37      5177
weighted avg       0.50      0.41      0.38      5177

KNN Classification Report :              precision    recall  f1-score   support

           1       0.61      0.47      0.53      2059
           2       0.56      0.65      0.61      2481
           3       0.29      0.33      0.31       637

   micro avg       0.54      0.54      0.54      5177
   macro avg       0.49      0.48      0.48      5177
weighted avg       0.55      0.54      0.54      5177

CART Classification Report :              precision    recall  f1-score   support

           1       0.60      0.64      0.62      2059
           2       0.67      0.39      0.50      2481
           3       0.25      0.59      0.35       637

   micro avg       0.52      0.52      0.52      5177
   macro avg       0.50      0.54      0.49      5177
weighted avg       0.59      0.52      0.53      5177
```

Figure 37: Classification Report

Figure 38 shows the web-based application and can be visited at link [15]

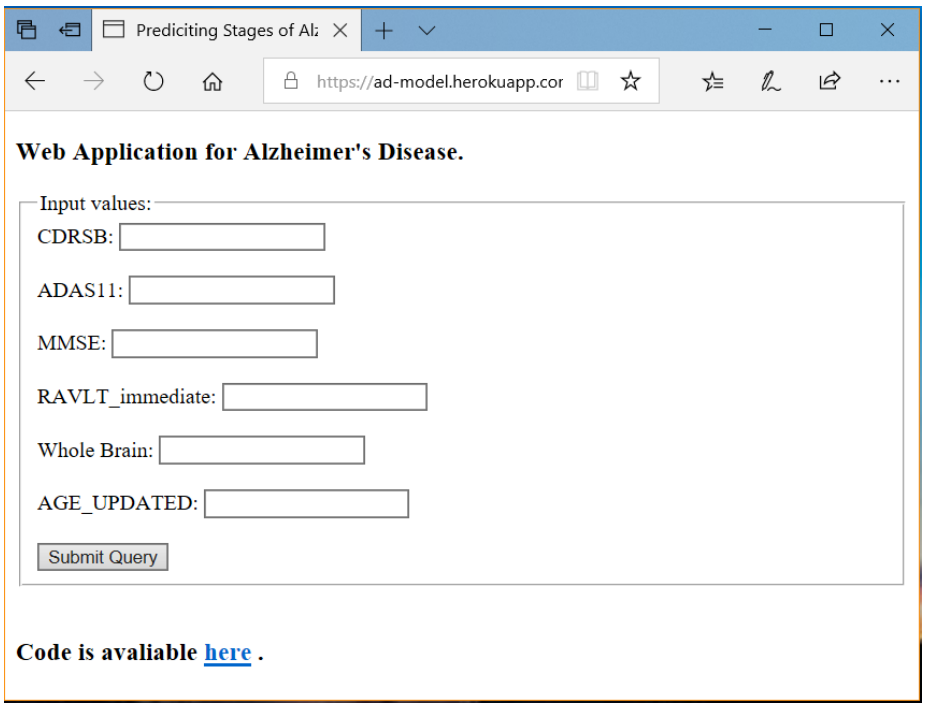


Figure 38: Web-based Application

10 Plots Generated During Data Analysis

Figure 39, Figure 40, Figure 41, Figure 42, Figure 43, Figure 44 and Figure 45 shows plots generated during data analysis.

[15] `https://ad-model.herokuapp.com/`

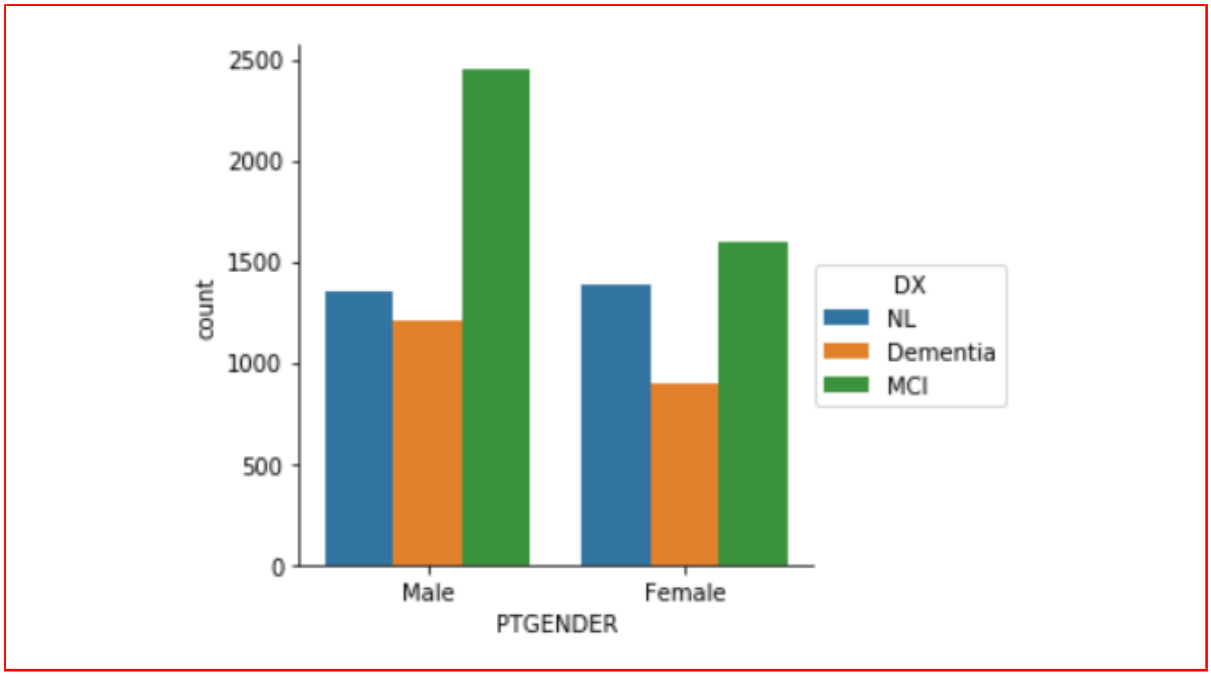


Figure 39: Images Generated During Data Analysis (1)

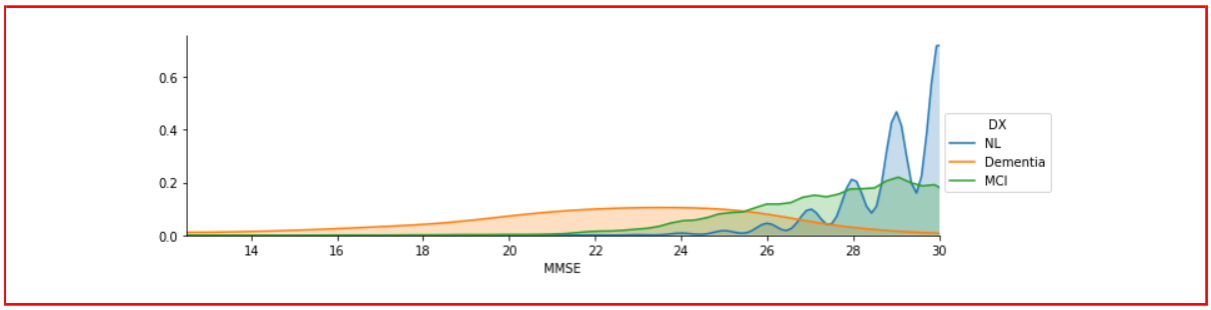


Figure 40: Images Generated During Data Analysis (2)

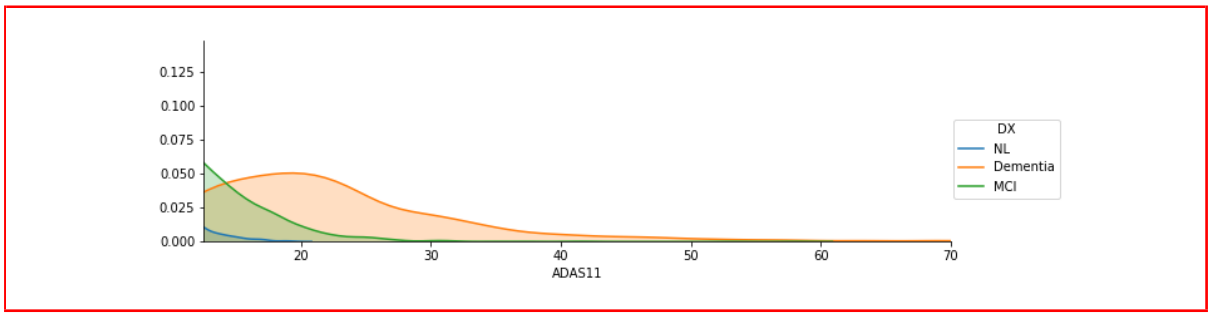


Figure 41: Images Generated During Data Analysis (3)

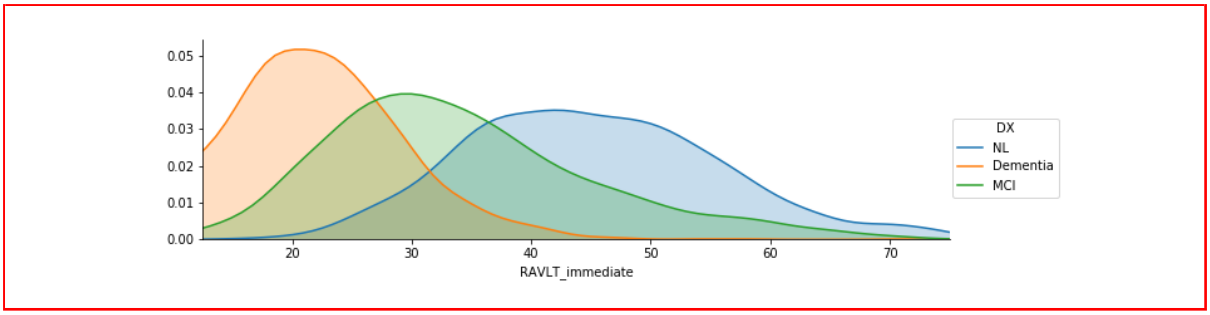


Figure 42: Images Generated During Data Analysis (4)

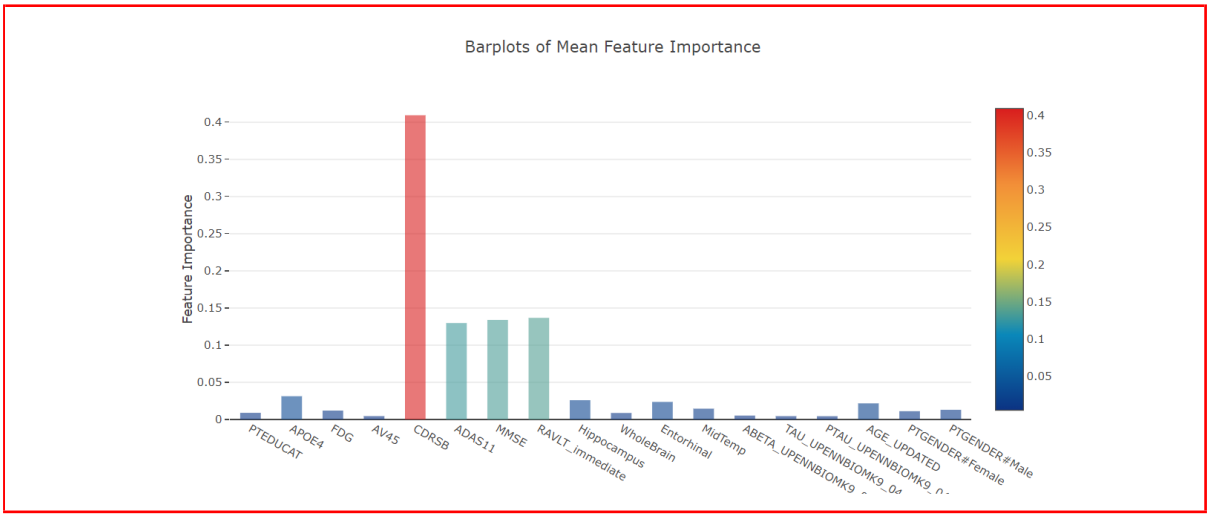


Figure 43: Images Generated During Data Analysis (5)

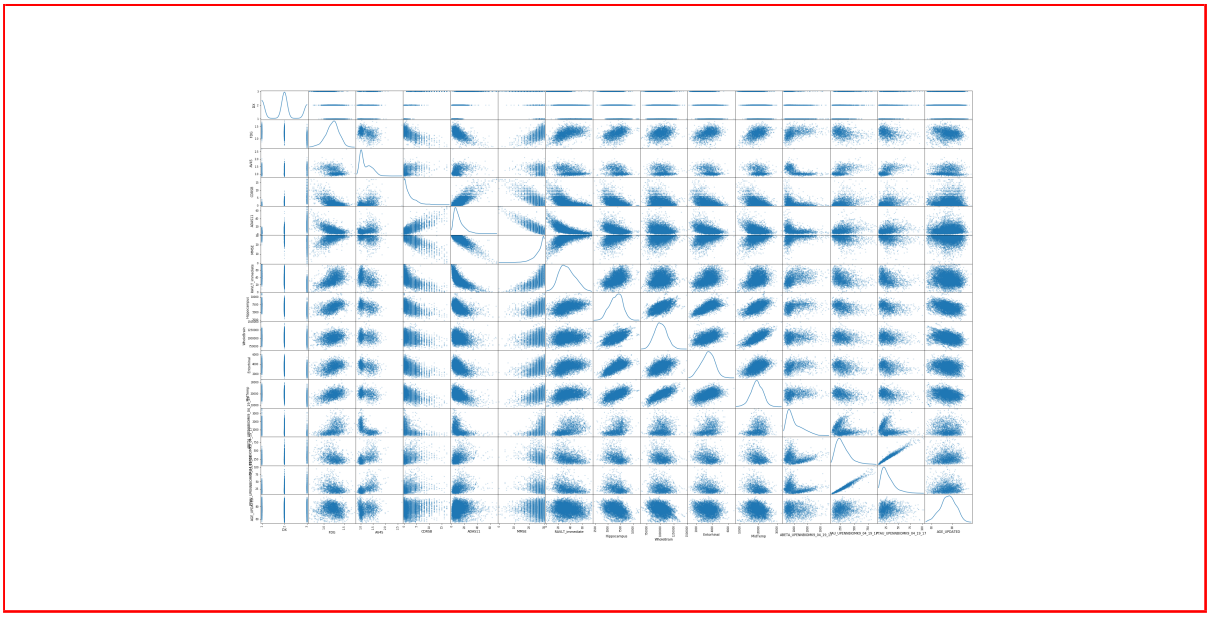


Figure 44: Images Generated During Data Analysis (6)

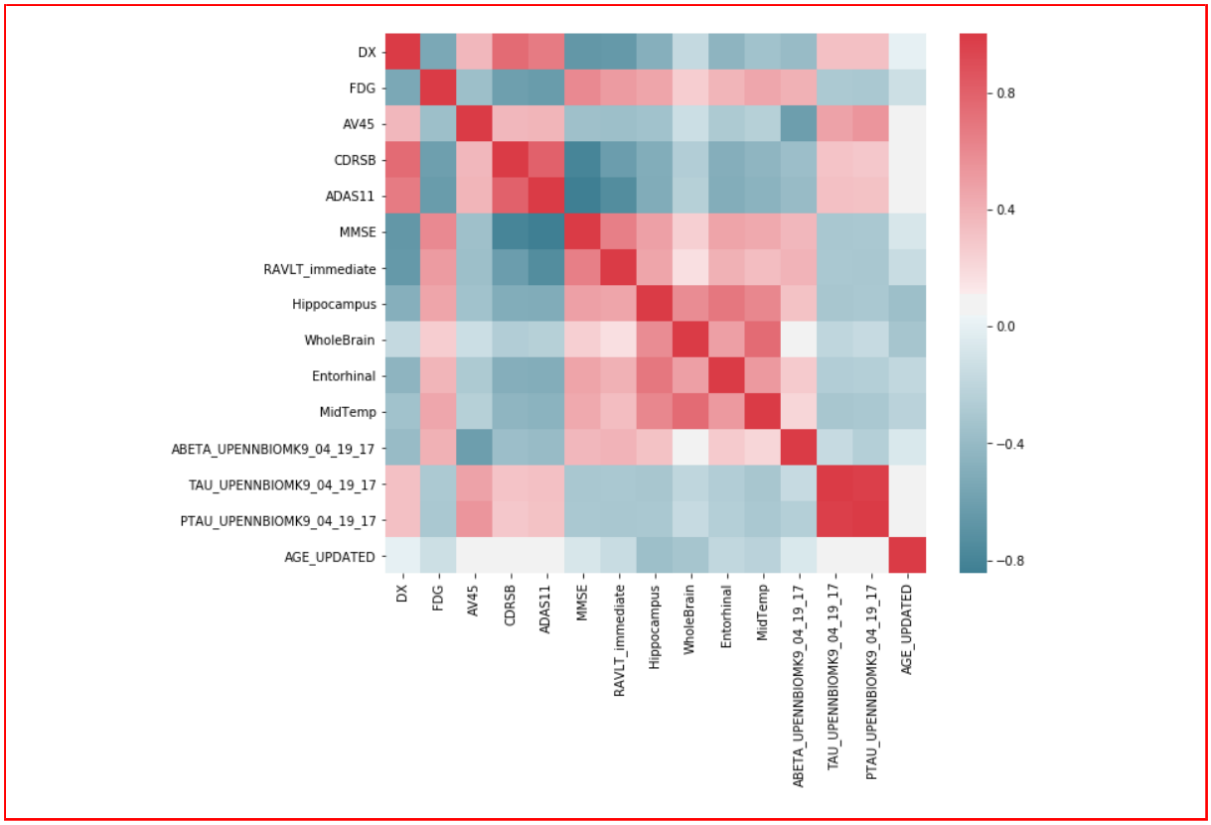


Figure 45: Images Generated During Data Analysis (7)