1

DATA SCIENCE

1.1 MRI AND ALZHEIMER

The `https://www.oasis-brains.org/`OASIS provides free datasets of MRI scans done on demented and non-demented people. They provide two free datasets containing demographic data accompanied with MRI scans. In this section we explore the demographic datasets OASIS-1 and OASIS-2[1, 2].

1.1.1 *Initial data exploration*

First we look at the structure of the data. The longitudinal set contains 373 scans and 15 rows an the cross-sectional set contains 436 scans and 12 rows. We plot the columns and data-types in 1. The length of the bar indicates the non-empty elopements. We observe that the longitudinal contains fewer empty elements than the cross-sectional set, hence we choose to start with this dataset for our regression and classification.

Subsequently, we check if there are easy to find correlations between the numerical columns. This is done by creating a correlation matrix heatmap seen in fig2. We want to attempt to predict the Clinical Dementia Rating or if a person is demented from this dataset, hence we look for correlations with the CDR rating. In both datasets it can be observed that there is an anti-correlation between

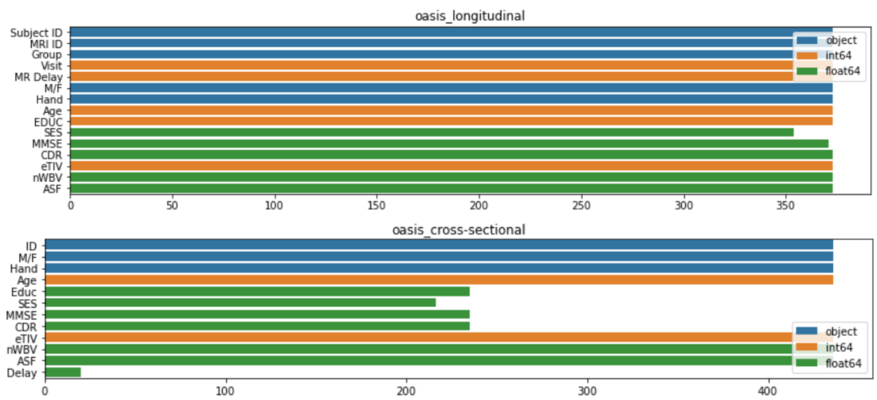
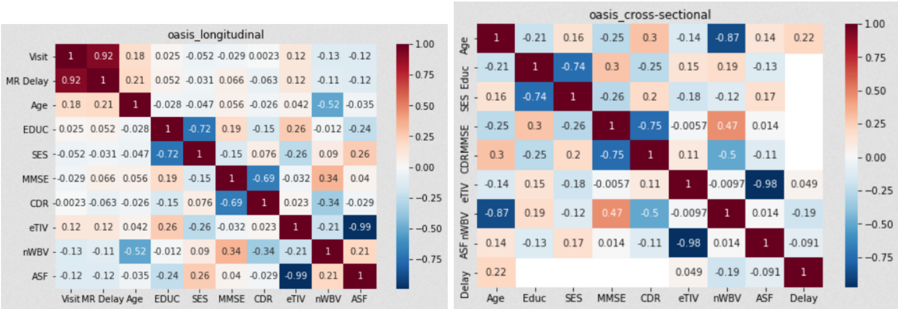


Figure 1: Colour shows the datatype, barlength shows non-empty elements

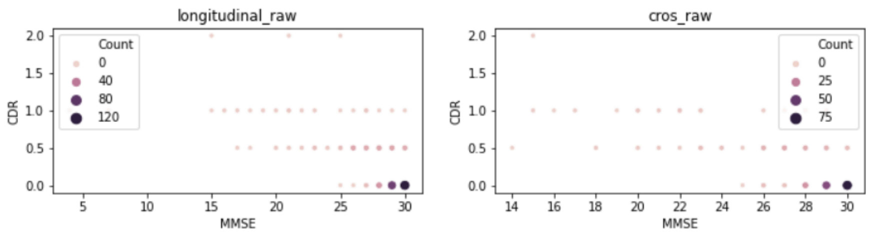


(a) Correlation for Longitudal (b) Correlation for Cross-sectional

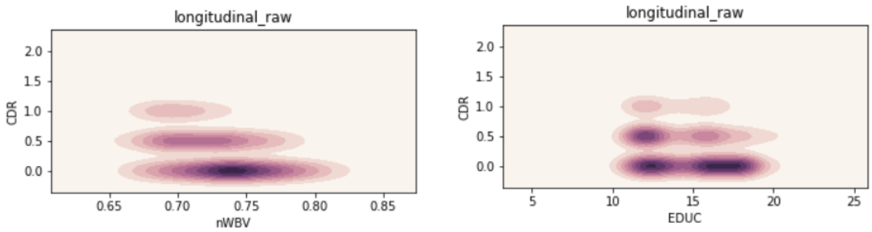
Figure 2: Dark red is highly correlated, white is no correlation, and blue is anti-correlated.

Mini Mental State Examination(MMSE) and Clinical Dementia Rating(CDR). We also observe small correlation with Normalize Whole Brain Volume(nWBV) and Education.

We visualize the correlations observed in fig3. The scatterplot in fig3a shows the correlation between CDR and MMSE. In fig3b and fig3c we visualise the correlation for nBWV and Education respectively.



(a) Scatterplot shopwing the relation between MMSE and CDR.



(b) Density plots for nBWV. *(c) Density plots for Educ.*

Figure 3: Visualization of correlations

1.1.2 *Regression and Classification*

Next we are going to see if we can predict the CDR using two machine learning techniques; regression and classification. We start out with training a XGBregressor using the features: Male/Female, Age, Education, nBWV, and MMSE. We train and test the regressor on differ-

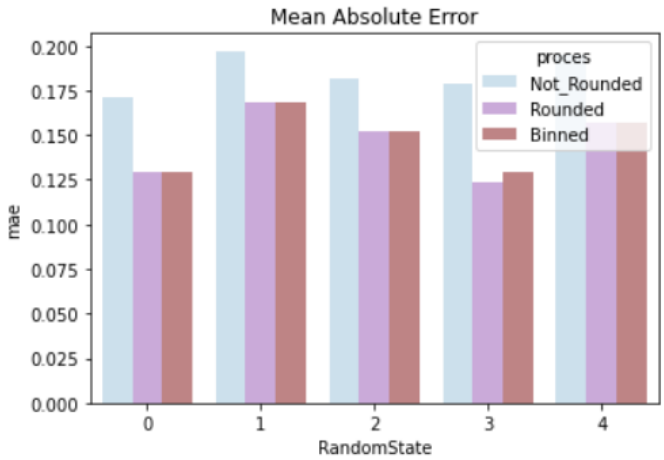
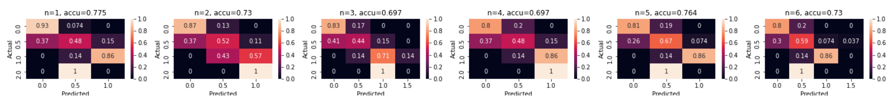
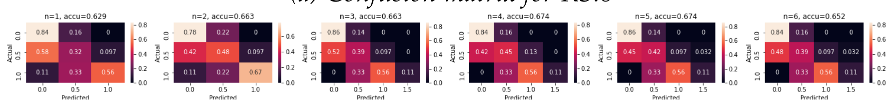


Figure 4: Comparing the MAE for not rounded, rounded and binned regressor results

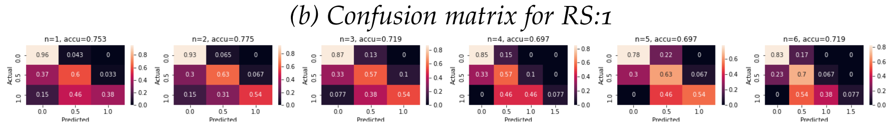
ent data splits. The regressor returns a float indicating the predicted CDR, however CDR rating is bracketed in 0, 0.5, 1, and 2. Therefor we apply three post-processes, i.e, no post-processing (Not rounded), rounding the data, and binning the data. In fig4 we compare the mean absolute error(mae) of these different processes. We conclude that using rounded and binned is superior to not rounding when comparing the mae.



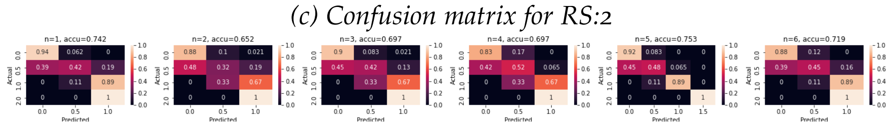
(a) Confusion matrix for RS:0



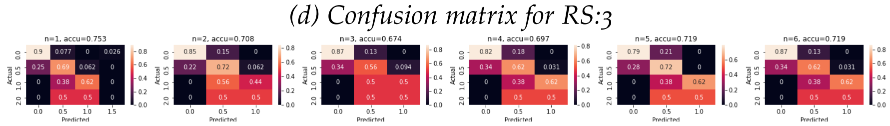
(b) Confusion matrix for RS:1



(c) Confusion matrix for RS:2



(d) Confusion matrix for RS:3

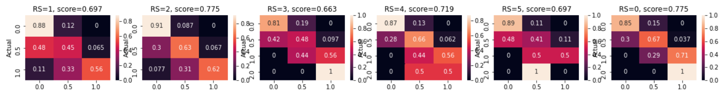


(e) Confusion matrix for RS:4

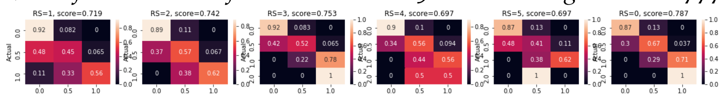
Figure 5: Confusion matrices for binned XGBregression.

In fig5 the binned result confusion matrices have been plotted for different data-splits and max depth. It can be observed that the confusion is rather dependent on the data splits, that is, it is very dependent on the supplied data.

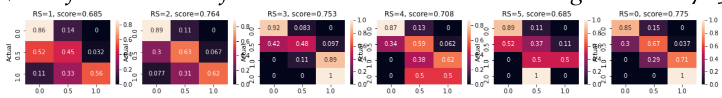
Next we look at a classifier instead of a regressor. In contrast to a regressor, which returns a continues float, a classifier classifies. Hence, we convert the CDR to categories. In this case we'll use the Random Forest Classifier with different number of estimators and compare the results. The confusion matrices for the RFC are plotted in fig6.



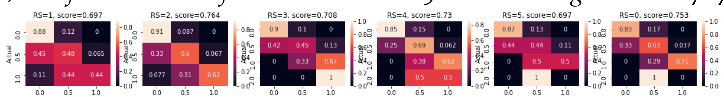
(a) Confusion matrix for n_estimators=50 with average error=0.777



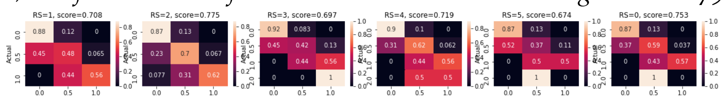
(b) Confusion matrix for n_estimators=100 with average error=0.785



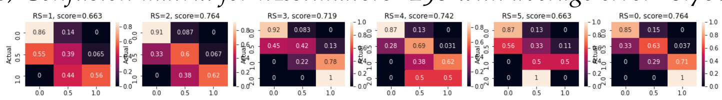
(c) Confusion matrix for n_estimators=150 with average error=0.787



(d) Confusion matrix for n_estimators=200 with average error=0.79



(e) Confusion matrix for n_estimators=250 with average error=0.788



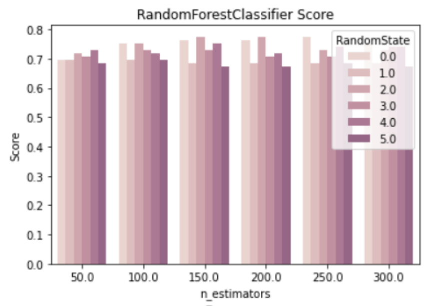
(f) Confusion matrix for n_estimators=300 with average error=0.788

Figure 6: Confusion matrices for RandomForestClassifier.

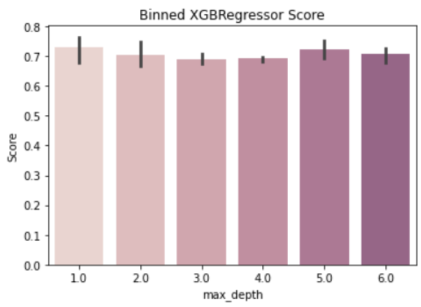
We observe that in general most confusion is with neighboring categories except in splitting using random state 4 and 5. Again this is more split than model dependent. In all splits there are only one, two ore three rows with a CDR of 2, hence they are hard to predict.

In fig7 we plot the score for the different models and Random states. Both models have similar results with a score around 0.7. Tuning the two variables explored here to obtain higher average scores also resulted in a larger confidence intervals.

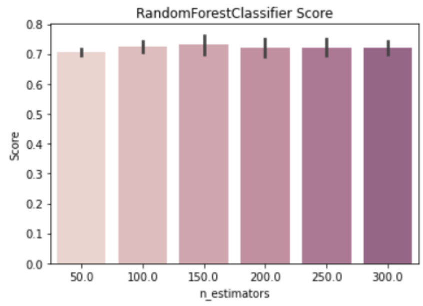
(a) XGBregression score for different max depths on different data splits.



(b) Random forest classifier score for different n on different data-splits

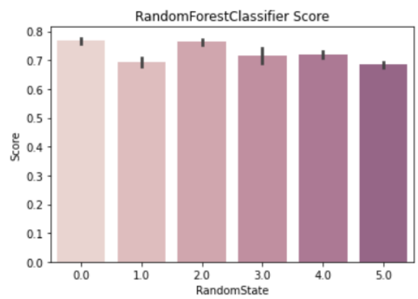


(c) XGBregression score for different max depths on different data splits.



(d) Random forest classifier score for different n on different data-splits

(e) XGBregression score for different random state splits.



(f) Random forest classifier score for different random state splits

Figure 7: Comparing regression and classifier for different n and RS

1.1.3 *Notes for the future*

There are a few takeaways for future projects. Considering the data used it would be great to have a larger sample size, especially for categories with few samples.

Looking at the data representation and exploration, I have the following notes

- Having more consistency in explored model variables.
- Having more consistency in representation of the data.
- Having more colour consistency.
- Creating confusion matrices averaged on different splittings.
- Creating barplots with barcolours dependent on their values.
- Creating a heatmap showing scores depending on RS and variables.
- Showing feature importance for the different models.

1.1.4 Code snippets

Code: Data exploration

```
def data_analysis_plot(data):
    #
    length = data.shape[1]
    lengthplot  = length/5
    plt.figure(figsize=(14,lengthplot))

    ax = sns.barplot(y=data.columns,x=data.notnull().sum(),
    hue=data.dtypes,dodge=False)
    ax.set_title(data.name)

    plt.figure(figsize=(8,5))
    ax2 = sns.heatmap(data.corr(),annot=True,cmap='RdBu_r')
    ax2.set_title(data.name)

    plt.figure(figsize=(8,5))
    ax3 = sns.heatmap(data.corr()[["CDR"]],annot=True,cmap='
    RdBu_r')
    ax3.set_title(data.name)

def data_info(data):
    print("Shape: ")
    print(data.shape)
    print("Columns: ")
    print(data.columns)

sets = [longitudinal_raw,cros_raw]
for datasets in sets:
    print(datasets.name)
    data_info(datasets)
    data_analysis_plot(datasets)
```

Code: Data Scatterplot MMSE-CDR

```
fused = longitudinal_raw.groupby(["MMSE","CDR"]).count()
fused["Count"]=fused["MRI ID"]
fused.head()
fused2 = cros_raw.groupby(["MMSE","CDR"]).count()
fused2["Count"]=fused2["ID"]
fused.head()
figz, axs = plt.subplots(ncols=2,figsize=(12,2.5))
sns.scatterplot(x=fused.index.get_level_values("MMSE"),y=fused
    .index.get_level_values("CDR"),data=fused,size="Count",
    hue="Count",ax=axs[0]).set_title("longitudinal_raw")
sns.scatterplot(x=fused2.index.get_level_values("MMSE"),y=
    fused2.index.get_level_values("CDR"),data=fused2,size="
    Count", hue="Count",ax=axs[1]).set_title("cros_raw")
```

Code: Regression

```
def multi_plot(y_test,prediction,n,vari):
    accu = np.round(accuracy_score(y_test, prediction),3)
    conf = pd.crosstab(y_test, prediction, rownames=['Actual'
    ], colnames=['Predicted'],normalize='index')
```

```
    #conf = confusion_matrix(y_test,prediction,normalize='true')
    sns.heatmap(conf, annot=True,ax=multi_ax[n-1]).set(xlabel='Predicted', ylabel='Actual',title=vari+"="+str(n)+", "+" score="+accu.astype('str'))
    #plot_confusion_matrix(model,X_test, y_test,normalize='true')
    return accu

def test_model2(X,y,n,rs):
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=rs)
    model = XGBRegressor(random_state=0,max_depth=n)
    model.fit(X_train,y_train)
    prediction = model.predict(X_test)
    rounded_pred = np.absolute(np.round(2*prediction)/2)
    mae = mean_absolute_error(rounded_pred,y_test)
    score = multi_plot(y_test.astype(str),rounded_pred.astype(str),n,vari="n")
    return score

#Create plots
l=7

reg_score = pd.DataFrame(columns=["max_depth","RandomState","Score"])

for i in range(5):
    rs = i
    multi_fig, multi_ax = plt.subplots(ncols=l-1,figsize=(l*3,2))
    multi_fig.tight_layout()
    for n in range(1,l):
        score = test_model2(X,y,n,rs)
        reg_score.loc[str(n)+str(rs)]=[n,rs,score]
    #output = multi_fig
    name = 'PP_dataS_MRIalz_RS'+str(i)+'.png'
    multi_fig.savefig(name)
    print(r"\begin{subfigure}{\textwidth}\includegraphics[width=\textwidth]{Pictures/datas/"+name+
"}\caption{Confusion matrix for RS:"+str(i)+"}\end{subfigure}")
```

Code: Classification

```
#GradientBoostingClassifier,RandomForestClassifier,
def classie(X,y,n,rs):
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=rs)
    nn = n*50
    model = RandomForestClassifier(n_estimators=nn)
    model.fit(X_train,y_train)
    prediction = model.predict(X_test)
    multi_plot(y_test,prediction,rs,vari="RS")
    accu = accuracy_score(y_test, prediction)
    return accu


```

```

X = longitudinal[features]
#y_g = longitudinal["Group"]
y_g = longitudinal["CDR"].astype('str')

l=7
nplots = 6

classi_error = pd.DataFrame(columns=["n_estimators","RandomState","Score"])
for n in range(1,l):
    multi_fig, multi_ax = plt.subplots(ncols=nplots,figsize=(l*3,2))
    #multi_fig.tight_layout()
    multi_fig.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=1, hspace=None)
    errtot = 0
    for rs in range(nplots):
        c_score = classie(X,y_g,n,rs)
        errtot+= c_score
        classi_error.loc[str(n)+str(rs)]=[n*50,rs,c_score]
    err = errtot/nplots
    #print(err)
    name = 'PP_dataS_MRIalz_RFCn='+str(n)+'.png'
    multi_fig.savefig(name)
    print(r"\begin{subfigure}{\textwidth}\includegraphics[width=\textwidth]{Pictures/datas/"+name+
r"}\caption{Confusion matrix for n{\textunderscore}estimators="+str(n*50)+" with average error="+str(round(err,3))+"}\end{subfigure}")
```

Code: Full code

```
# %% [code]
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

```

```
# You can write up to 5GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

print("Setup done")

# %% [code]
long_path = "/kaggle/input/mri-and-alzheimers/oasis_longitudinal.csv"
longitudinal_raw = pd.read_csv(long_path)
longitudinal_raw.name = "oasis_longitudinal"

cross_path = "/kaggle/input/mri-and-alzheimers/oasis_cross-sectional.csv"
cros_raw = pd.read_csv(cross_path)
cros_raw.name = "oasis_cross-sectional"
print("Datasets done")

# %% [code]
def data_analysis_plot(data):
    #
    length = data.shape[1]
    lengthplot = length/5
    plt.figure(figsize=(14,lengthplot))

    ax = sns.barplot(y=data.columns,x=data.notnull().sum(),hue=data.dtypes,dodge=False)
    ax.set_title(data.name)

    plt.figure(figsize=(8,5))
    ax2 = sns.heatmap(data.corr(),annot=True,cmap='RdBu_r')
    ax2.set_title(data.name)

    plt.figure(figsize=(8,5))
    ax3 = sns.heatmap(data.corr()[["CDR"]],annot=True,cmap='RdBu_r')
    ax3.set_title(data.name)

def data_info(data):
    print("Shape: ")
    print(data.shape)
    print("Columns: ")
    print(data.columns)

sets = [longitudinal_raw,cros_raw]
for datasets in sets:
    print(datasets.name)
    data_info(datasets)
    data_analysis_plot(datasets)


# %% [code]
meaning = {"SES":"Socioeconomic Status","MMSE":"Mini Mental State Examination","CDR":"Clinical Dementia Rating",
```

```
        "eTIV":"Estimated Total intracranial Voluma","nWBV":"Normalize Whole Brain Volume","ASF":"Atlas Scaling Factor"}





# %% [code]
def obj_cols(data):
    s = (data.dtypes == 'object')
    object_cols = list(s[s].index)
    print("Dataset: "+data.name)
    for i in object_cols:
        if ("ID" in i) != True:
            print(i+" unique values:")
            print(data[i].value_counts())
    return object_cols

for data in sets:
    obj_cols(data)

# %% [code]
#Find dupliacte IDs
def non_uniq(data):
    print(data.value_counts()[data.value_counts().values > 1])

for i in [longitudinal_raw["Subject ID"],longitudinal_raw["MRI ID"],cros_raw["ID"]]:
    non_uniq(i)
cros_raw = cros_raw.rename(columns={"Educ":"EDUC"})
cros_raw.columns

# %% [code]
corcols = ["MMSE","nWBV","EDUC"]
for i in corcols:
    fused = longitudinal_raw.groupby([i,"CDR"]).count()
    fused["Count"]=fused["MRI ID"]
    fused.head()
    fused2 = cros_raw.groupby([i,"CDR"]).count()
    fused2["Count"]=fused2["ID"]
    fused.head()
    figz, axs = plt.subplots(ncols=2,figsize=(12,2.5))
    sns.scatterplot(x=fused.index.get_level_values(i),y=fused.index.get_level_values("CDR"),data=fused,size="Count", hue="Count",ax=axs[0]).set_title("longitudinal_raw")
    sns.scatterplot(x=fused2.index.get_level_values(i),y=fused2.index.get_level_values("CDR"),data=fused2,size="Count", hue="Count",ax=axs[1]).set(title="cros_raw")

# %% [code]
for i in corcols:
    cm = sns.cubehelix_palette(light=1, as_cmap=True)
    figz2, axss = plt.subplots(ncols=2,figsize=(12,2.5))
    sns.kdeplot(longitudinal_raw[i], longitudinal_raw["CDR"], shade=True,cmap=cm ,ax=axss[0]).set_title("longitudinal_raw")
```

```
    sns.kdeplot(cros_raw[i], cros_raw["CDR"],shade=True,cmap=cm ,ax=axss[1]).set_title("cros_raw")

# %% [code]
longitudinal = longitudinal_raw.copy()
longitudinal["M/F"]=longitudinal["M/F"].map({"M":0,"F":1})
print(longitudinal.head())
print(longitudinal.shape)
longitudinal = longitudinal.dropna()
print(longitudinal.shape)

# %% [code]
print("Import ML modules")
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import accuracy_score
from xgboost import XGBRegressor
print("Importing done")

# %% [code]
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline

featuress=["M/F","Age","EDUC","nWBV","MMSE","eTIV"]
features=featuress[0:5]
print("Features used:")
print(features)
X = longitudinal[features]
y = longitudinal["CDR"]

def get_score(line):
    # Multiply by -1 since sklearn calculates *negative* MAE
    scores = -1 * cross_val_score(line, X, y,
                                  cv=5,
                                  scoring='neg_mean_absolute_error')
    print("Average MAE score:", scores.mean())
    return scores.mean()

def runmodels(X,y):
    rf_model = RandomForestRegressor(random_state=1, n_estimators=100)
    XGB_model = XGBRegressor(random_state=0)
    models = [rf_model,XGB_model]
    for model in models:
        my_pipeline = Pipeline(steps=[
                ('model', model)
                ])
        get_score(my_pipeline)

runmodels(X,y)

# %% [code]
prediction = [0,0.4,1.1,1.4,2,2.5]
binned_pred = pd.cut(prediction,[-1,0.25,0.75,1.5,10],labels=[0,0.5,1,2])
```

```
print(binned_pred)
print(binned_pred.astype('str'))

# %% [code]
def show_accuracy(y_test,prediction,vari,n):
    score = accuracy_score(y_test, prediction)
    print(score)
    conf = pd.crosstab(y_test, prediction, rownames=['Actual'], colnames=['Predicted'],normalize='index')
    #conf = confusion_matrix(y_test,prediction,normalize='true')
    figy , axconf = plt.subplots(figsize=(5,2))
    tit = vari+": "+str(n)+", score:"+str(round(score,3))
    axconf = sns.heatmap(conf, annot=True).set(xlabel='Predicted', ylabel='Actual',title=tit)
    #plot_confusion_matrix(model,X_test, y_test,normalize='true')
    return accuracy_score

def test_model(X,y,n):
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=n)
    XGB_model = XGBRegressor(random_state=0,max_depth=5 )
    XGB_model.fit(X_train,y_train)
    prediction = XGB_model.predict(X_test)
    mae1 = mean_absolute_error(prediction,y_test)
    print("Not rounded: ",mae1)
    rounded_pred = np.absolute(np.round(2*prediction)/2)
    mae2 = mean_absolute_error(rounded_pred,y_test)
    print("Rounded: ",mae2)
    binned_pred = pd.cut(prediction,[-1,0.25,0.75,1.5,10], labels=[0,0.5,1,2],right=False)
    mae3 = mean_absolute_error(binned_pred,y_test)
    print("Binned: ",mae3)
    vari = "Split random state"
    show_accuracy(y_test.astype(str),binned_pred.astype(str), vari,n)
    return [mae1,mae2,mae3]

regr_mae = pd.DataFrame(columns=["RandomState","proces","mae"])
proc=["Not_Rounded","Rounded","Binned"]
for n in range(5):
    result=test_model(X,y,n)
    for i in range(len(result)):
        regr_mae.loc[str(n)+str(i)]=[n]+[proc[i]]+[result[i]]
print(regr_mae)
plt.show()
sns.barplot(x="RandomState", y="mae",data=regr_mae,hue="proces", palette=sns.light_palette("green")).set(title="Mean Absolute Error")

# %% [code]
plt.show()
#clors = sns.color_palette("RdYlGn", 5)
clors = sns.color_palette("cubehelix_r",6)
```

```
sns.barplot(x="RandomState", y="mae",data=regr_mae,hue="proces
    ", palette=clors).set(title="Mean Absolute Error")

# %% [code]
from sklearn.preprocessing import FunctionTransformer
XGB_model = XGBRegressor(random_state=0)
trans = FunctionTransformer(np.round, validate=True)
le_line = Pipeline(steps=[('Custom transformation',trans),
                  ('model', XGB_model)]
                  )

scores = -1 * cross_val_score(le_line, X, y,
                              cv=5,
                              scoring='
    neg_mean_absolute_error')
print(scores)
print("Average MAE score:", scores.mean())

# %% [code]
print("Doing Classifiers")
from sklearn.ensemble import GradientBoostingClassifier,
    RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix


def classifiers(longitudinal):
    featuress=["M/F","Age","EDUC","nWBV","MMSE","eTIV"]
    features=featuress[0:5]
    print(features)
    X = longitudinal[features]
    y = longitudinal["CDR"]
    y_trans = y.copy().astype(str)
    X_train, X_test, y_train, y_test = train_test_split(X,
    y_trans, random_state=1)

    #GBC = GradientBoostingClassifier(n_estimators=200)
    #RFC = RandomForestClassifier(n_estimators = 200)
    #classifiers = [GBC,RFC]



    for n in range(2,3):
        vari = "n_estimators"
        esti = n*50
        model = RandomForestClassifier(n_estimators=esti)
        model.fit(X_train,y_train)
        prediction = model.predict(X_test)
        show_accuracy(y_test,prediction,vari,n)
        print(model.feature_importances_)



classifiers(longitudinal)


# %% [code]
```

```
def multi_plot(y_test,prediction,n,vari):
    accu = np.round(accuracy_score(y_test, prediction),3)
    conf = pd.crosstab(y_test, prediction, rownames=['Actual'], colnames=['Predicted'],normalize='index')
    #conf = confusion_matrix(y_test,prediction,normalize='true')
    sns.heatmap(conf, annot=True,ax=multi_ax[n-1]).set(xlabel='Predicted', ylabel='Actual',title=vari+"="+str(n)+", "+"score="+accu.astype('str'))
    #plot_confusion_matrix(model,X_test, y_test,normalize='true')
    return accu

def test_model2(X,y,n,rs):
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=rs)
    model = XGBRegressor(random_state=0,max_depth=n)
    model.fit(X_train,y_train)
    prediction = model.predict(X_test)
    rounded_pred = np.absolute(np.round(2*prediction)/2)
    mae = mean_absolute_error(rounded_pred,y_test)
    score = multi_plot(y_test.astype(str),rounded_pred.astype(str),n,vari="max depth")
    return score

#Create plots
l=7

reg_score = pd.DataFrame(columns=["max_depth","RandomState","Score"])

for i in range(5):
    rs = i
    multi_fig, multi_ax = plt.subplots(ncols=l-1,figsize=(l*3,2))
    multi_fig.tight_layout()
    for n in range(1,l):
        score = test_model2(X,y,n,rs)
        reg_score.loc[str(n)+str(rs)]=[n,rs,score]
    #output = multi_fig
    name = 'PP_dataS_MRIalz_RS'+str(i)+'.png'
    multi_fig.savefig(name)
    print(r"\begin{subfigure}{\textwidth}\includegraphics[width=\textwidth]{Pictures/datas/"+name+
"}\caption{Confusion matrix for RS:"+str(i)+"}\end{subfigure}")


# %% [code]
sns.barplot(x=reg_score["max_depth"],y=reg_score["Score"],
                hue=reg_score["RandomState"],palette=sns.cubehelix_palette(n_colors=10)).set(title="Binned XGBRegressor Score")

# %% [code]
sns.barplot(x=reg_score["RandomState"],y=reg_score["Score"],
```

```
            hue=reg_score["max_depth"],palette=sns.cubehelix_palette(n_colors=10)).set(title="Binned XGBRegressor Score")

# %% [code]
#GradientBoostingClassifier,RandomForestClassifier,
def classie(X,y,n,rs):
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=rs)
    nn = n*50
    model = RandomForestClassifier(n_estimators=nn)
    model.fit(X_train,y_train)
    prediction = model.predict(X_test)
    multi_plot(y_test,prediction,rs,vari="RS")
    accu = accuracy_score(y_test, prediction)
    return accu



X = longitudinal[features]
#y_g = longitudinal["Group"]
y_g = longitudinal["CDR"].astype('str')

l=7
nplots = 6

classi_error = pd.DataFrame(columns=["n_estimators","RandomState","Score"])
for n in range(1,l):
    multi_fig, multi_ax = plt.subplots(ncols=nplots,figsize=(l*3,2))
    #multi_fig.tight_layout()
    multi_fig.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=0.1, hspace=None)
    errtot = 0
    for rs in range(nplots):
        c_score = classie(X,y_g,n,rs)
        errtot+= c_score
        classi_error.loc[str(n)+str(rs)]=[n*50,rs,c_score]
    err = errtot/nplots
    #print(err)
    name = 'PP_dataS_MRIalz_RFCn='+str(n)+'.png'
    multi_fig.savefig(name)
    print(r"\begin{subfigure}{\textwidth}\includegraphics[width=\textwidth]{Pictures/datas/"+name+
r"}\caption{Confusion matrix for n{\textunderscore}estimators="+str(n*50)+" with average error="+str(round(err,3))+"}\end{subfigure}")

# %% [code]
classi_error.head()
sns.barplot(x=classi_error["n_estimators"],y=classi_error["Score"],
            hue=classi_error["RandomState"],palette=sns.cubehelix_palette(n_colors=10)).set(title="RandomForestClassifier Score")

```

```
# %% [code]
print(reg_score.groupby("max_depth").agg({"Score":["mean","std"]}))
print(reg_score.groupby("RandomState").agg({"Score":["mean","std"]}))
print(classi_error.groupby("n_estimators").agg({"Score":["mean","std"]}))
print(classi_error.groupby("RandomState").agg({"Score":["mean","std"]}))

sns.barplot(reg_score["max_depth"],reg_score["Score"],palette=sns.cubehelix_palette(n_colors=10)).set(title="Binned XGBRegressor Score")
plt.show()
sns.barplot(classi_error["n_estimators"],classi_error["Score"],palette=sns.cubehelix_palette(n_colors=10)).set(title="RandomForestClassifier Score")


# %% [code]
sns.barplot(reg_score["RandomState"],reg_score["Score"],palette=sns.cubehelix_palette(n_colors=10)).set(title="Binned XGBRegressor Score")
plt.show()
sns.barplot(classi_error["RandomState"],classi_error["Score"],palette=sns.cubehelix_palette(n_colors=10)).set(title="RandomForestClassifier Score")


# %% [code]
v = reg_score["Score"].values
colors=plt.cm.plasma((v-v.min())/(v.max()-v.min()))
sns.barplot(reg_score["max_depth"],reg_score["Score"],palette=colors).set(title="Binned XGBRegressor Score")

# %% [code]
def checksplit(n,X,y):
    for rs in range(5):
        X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=rs)
        plt.show()
        sns.countplot(x=y_train)
        print(y_train.value_counts())

checksplit(5,X,y)

# %% [code]
```

BIBLIOGRAPHY

[1] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner, "Open access series of imaging studies (oasis): Cross-sectional mri data in young, middle aged, nondemented, and demented older adults," *Journal of Cognitive Neuroscience*, vol. 19, no. 9, pp. 1498–1507, 2007.

[2] D. S. Marcus, A. F. Fotenos, J. G. Csernansky, J. C. Morris, and R. L. Buckner, "Open access series of imaging studies: Longitudinal mri data in nondemented and demented older adults," *Journal of Cognitive Neuroscience*, vol. 22, no. 12, pp. 2677–2684, 2010. PMID: 19929323.