Project1FINAL

February 28, 2025

1 Machine Learning in Python - Project 1

Otis Laundon, Kai Inamdar, Peter Bradshaw, and Ashe Raymond-Barker

[206]:
```
# Data libraries
import pandas as pd
import numpy as np

# Plotting libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting defaults
plt.rcParams['figure.figsize'] = (8,5)
plt.rcParams['figure.dpi'] = 80

# sklearn modules
import sklearn
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.model_selection import GridSearchCV, KFold, train_test_split
from sklearn.preprocessing import OneHotEncoder, OrdinalEncoder,␣
 ↪StandardScaler, FunctionTransformer
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.compose import make_column_transformer
from sklearn.compose import make_column_selector
from sklearn.linear_model import Lasso, Ridge
```

[ ]:
```
# Load data
d = pd.read_csv("adnidata.csv")
d.head()
```

[ ]:
```
   Unnamed: 0  RID  ADAS13.bl  ADAS13.m24   AGE DX.bl PTGENDER  PTEDUCAT  \
0           1    3      31.00       37.67  81.3    AD     Male        18
1           2    5      14.67       11.00  73.7    CN     Male        16
```

```
2            3   6      25.67        22.67  80.4  LMCI   Female         13
3            4   7      40.33        47.00  75.4    AD     Male         10
4            5  10      24.33        30.33  73.9    AD   Female         12

         PTETHCAT       PTRACCAT  PTMARRY  APOE4  Ventricles  Hippocampus  \
0  Not Hisp/Latino          White  Married    1.0     84599.0       5319.0
1  Not Hisp/Latino          White  Married    0.0     34062.0       7075.0
2  Not Hisp/Latino          White  Married    0.0     39826.0       5348.0
3      Hisp/Latino  More than one  Married    1.0     25704.0       6729.0
4  Not Hisp/Latino          White  Married    1.0     26820.0       5485.0

   WholeBrain  Entorhinal  Fusiform  MidTemp           ICV
0   1129834.0      1791.0   15506.0  18422.0  1.920691e+06
1   1116633.0      4433.0   24788.0  21614.0  1.640766e+06
2    927510.0      2277.0   17963.0  17802.0  1.485834e+06
3    875798.0      2050.0   12063.0  15374.0  1.353519e+06
4   1033542.0      2676.0   16761.0  19741.0  1.471184e+06
```

2 Introduction

This report aspires to produce an accurate model in predicting individual performance on the Alzheimer's Disease Assessment Scale-Cognitive Subscale (ADAS-Cog) 13, as taken 24 months after a baseline test. Early identification of the disease for those at risk would alleviate the pressure on healthcare, social services and individuals in dealing with a crippling and near-unforeseeable disorder.

Our model is based on a number of variables measured at the time of the baseline test. While the most telling predictor of future scores appears to be the baseline test, as expected, we have found many features to have nuanced but significant effect in predicting ADAS-Cog 13 scores. We built our final model as a function of baseline ADAS-Cog 13 score, Age, diagnosis, gender, years of education, number of APOE-4 alleles, and the volume of the subjects' hippocampus, entorhinal cortex, fusiform gyrus, middle temporal lobe and intracranial volume. We reduced variables by cross-validating penalty functions, and chose a final model that explained a maximum amount of variance we found in the data while consistently minimizing under-predictions to ensure that no individuals should slip through the cracks. While the model performs adequately against our training and testing data, the effect of individual measured characteristics appeared so nuanced that these analysts would recommend further study in other potential harbingers to find more direct causes of and hence indicators for the Alzheimer's disorder.

3 Exploratory Data Analysis and Feature Engineering

3.1 Dataset Overview

The dataset consists of 1038 observations of unique individuals. There are a range of features capturing demographic, cognitive, genetic, and MRI-captured information at a baseline visit, which we will use to predict **ADAS13.m24**, the ADAS-Cog 13 score at 24-month follow-up visit. The 16 features and target can be categorised as follows:

- **Cognitive Scores and Diagnosis:**
  - **ADAS13.bl** (Numeric): Baseline ADAS-Cog 13 score, assessing cognitive function at the start of the study.
  - **ADAS13.m24** (Numeric): 24-month follow-up ADAS-Cog 13 score, representing cognitive decline, the target of our analysis.
  - **DX.bl** (Ordinal): Diagnosis at baseline, with four categories: Cognitively Normal (CN), Early Mild Cognitive Impairment (ECMI), Late Cognitive Impairment (LCMI), and Alzheimer's Disease (AD).
- **Demographics:**
  - **AGE** (Numeric): Age of the individual at baseline.
  - **PTGENDER** (Binary Categorical): Gender of the individual (male or female).
  - **PTEDUCAT** (Ordinal): Number of years of formal education.
  - **PTETHCAT** (Categorical): Ethnicity of the individual (Not Hispanic/Latino, Hispanic/Latino, or Unknown).
  - **PTRACCAT** (Categorical): Race of the individual (7 levels, white, black, asian, american indian/alaskan, hawaiian/other pacific islands, more than one, unknown).
  - **PTMARRY** (Categorical): Marital status at baseline.
- **Genetic Information:**
  - **APOE4** (Ordinal): APOE genotype indicating the number of copies of the e4 allele (0,1, or 2).
- **MRI/Neuroimaging Features At Baseline:**
  - **Ventricles** (Numeric): Volume of the ventricles in cubic millimeters.
  - **Hippocampus** (Numeric): Volume of the hippocampus in cubic millimeters.
  - **WholeBrain** (Numeric): Volume of the whole brain in cubic millimeters.
  - **Entorhinal** (Numeric): Volume of the entorhinal cortex in cubic millimeters.
  - **Fusiform** (Numeric): Volume of the fusiform gyrus in cubic millimeters.
  - **MidTemp** (Numeric): Volume of the middle temporal lobe in cubic millimeters.
  - **ICV** (Numeric): Intracranial volume in cubic millimeters.

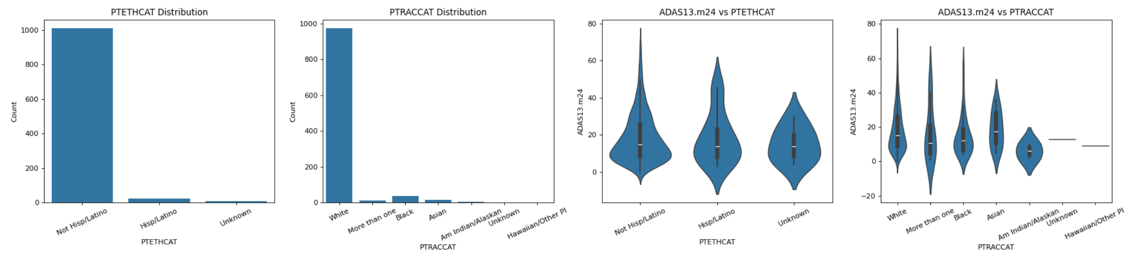
3.2 Null Values

A total of 153 entries contain missing data, primarily due to individuals lacking MRI values. Upon comparing the null and non-null populations, we found no significant differences between the two. As a result, we decided to remove the samples with missing values, reducing our sample size to 885.

3.3 Dropping features

As seen in the figures below, the features **PTRACCAT** and **PTETHCAT** are highly imbalanced, with only 27 individuals outside the Not Hispanic/Latino group and 65 individuals outside the White category. While we could apply techniques such as oversampling minority groups or undersampling the majority class, these methods risk either introducing synthetic patterns that do not generalise, or discarding valuable data. Additionally, the violin plots below suggest no clear relationship between these features and the target variable, **ADAS13.m24**. Given the limited representation of minority groups, any patterns learned would likely be misleading rather than informative. Therefore, we exclude these two features to prevent the model from fitting spurious associations and avoid drawing unreliable conclusions.

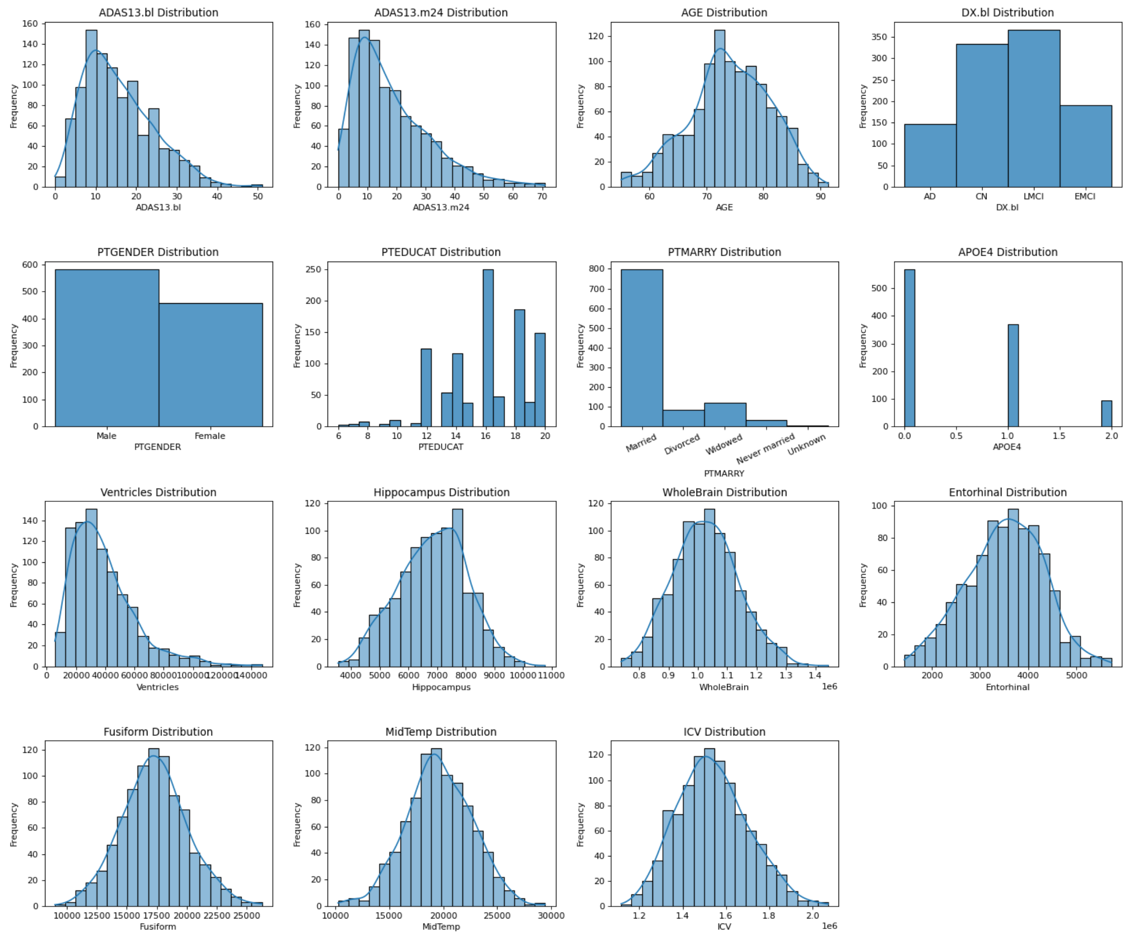
[208]:
```python
# Define the columns for histograms and violin plots
columns_to_display = ['PTETHCAT', 'PTRACCAT']

# Set the size of the plots
fig, axes = plt.subplots(1, 4, figsize=(22, 5))
axes = axes.flatten()

# Plot histograms in the columns_to_display row
for i, col in enumerate(columns_to_display):
    sns.countplot(data=d, x=col, ax=axes[i])
    axes[i].set_title(f'{col} Distribution')
    axes[i].set_ylabel('Count')
    axes[i].tick_params(axis='x', rotation=25)  # Rotate x-axis labels
    sns.violinplot(x=d[col], y=d["ADAS13.m24"], ax=axes[i + 2])
    axes[i + 2].set_title(f"ADAS13.m24 vs {col}")
    axes[i + 2].tick_params(axis='x', rotation=25)  # Rotate x-axis labels

# Adjust layout and show the plot
plt.tight_layout()
plt.show()
```



3.4 Data Structure and Preprocessing

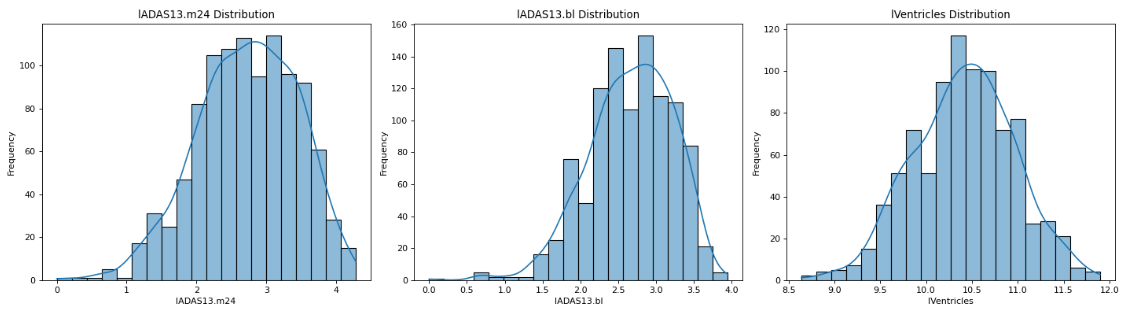
Now, we analyse the distributions of the features and the target variable, **ADAS13.m24**. By visualising the histograms of these variables, we can gain insights into their underlying distributions, identify any skewness, and assess the presence of outliers. The histograms also allow us to examine whether the data follows a roughly normal distribution or if transformations might be necessary for further analysis.

[209]:
```python
# Set the size of the plots
plt.figure(figsize=(18, 15))

ordinal_categorical_columns = ['PTGENDER','DX.bl','PTMARRY','PTEDUCAT','APOE4']

# Loop through the numeric columns to create a histogram for each
for i, col in enumerate(d.drop(columns=['Unnamed: 0','RID','PTETHCAT',␣
 ↪'PTRACCAT']).columns, 1):
    plt.subplot(4, 4, i)  # Adjust the number of rows/columns as per your data
    if col in ordinal_categorical_columns:
          sns.histplot(d[col], kde=False, bins=20)
    else:
      sns.histplot(d[col], kde=True, bins=20)  # KDE adds a smooth curve
    if col == 'PTMARRY':
      plt.tick_params(axis='x', rotation=25)  # Rotate x-axis labels
    plt.title(f'{col} Distribution')
    plt.xlabel(col)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



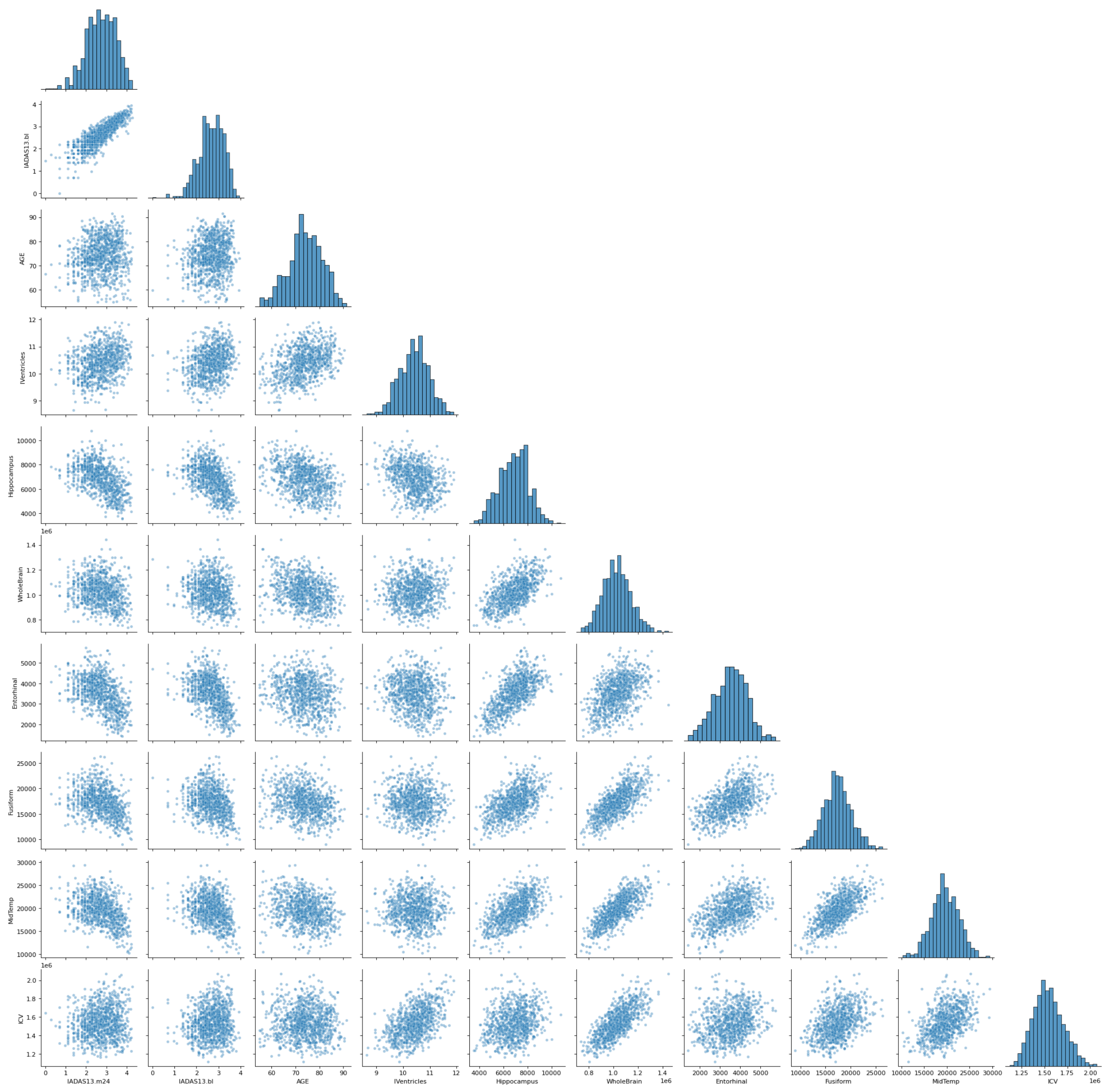
From these histograms, it is clear that the target, **ADAS13.m24**, as well as the features **ADAS13.bl** and **Ventricles** have significant positive skew. Thus, in our model, we have applied a natural logarithm transformation to these series in an effort to reduce the effect of extreme values and improve the performance of our model (specifically reducing heteroscedascicity). The below histograms demonstrate the skew-reducing effect of log-transforming these series:

```
[210]: # Set the size of the plots
plt.figure(figsize=(18, 5))

d[['lADAS13.m24','lADAS13.bl','lVentricles']] = np.log(d[['ADAS13.m24','ADAS13.
  ↪bl','Ventricles']] + 1)
columns_to_display = ['lADAS13.m24','lADAS13.bl','lVentricles']
# Loop through the numeric columns to create a histogram for each
for i, col in enumerate(columns_to_display, 1):
    plt.subplot(1, 3, i)  # Adjust the number of rows/columns as per your data
    sns.histplot(d[col], kde=True, bins=20)  # KDE adds a smooth curve
    plt.title(f'{col} Distribution')
    plt.xlabel(col)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```



We now use a pairplot to analyse correlations and relationships between different numerical features and the target:

```
[211]: numerical_columns = ["lADAS13.m24", "lADAS13.bl", "AGE", "lVentricles",␣
  ↪"Hippocampus", "WholeBrain", "Entorhinal", "Fusiform", "MidTemp",'ICV']
sns.pairplot(d[numerical_columns], corner=True,  plot_kws={'alpha': 0.4, 's':␣
  ↪20})
plt.show()
```

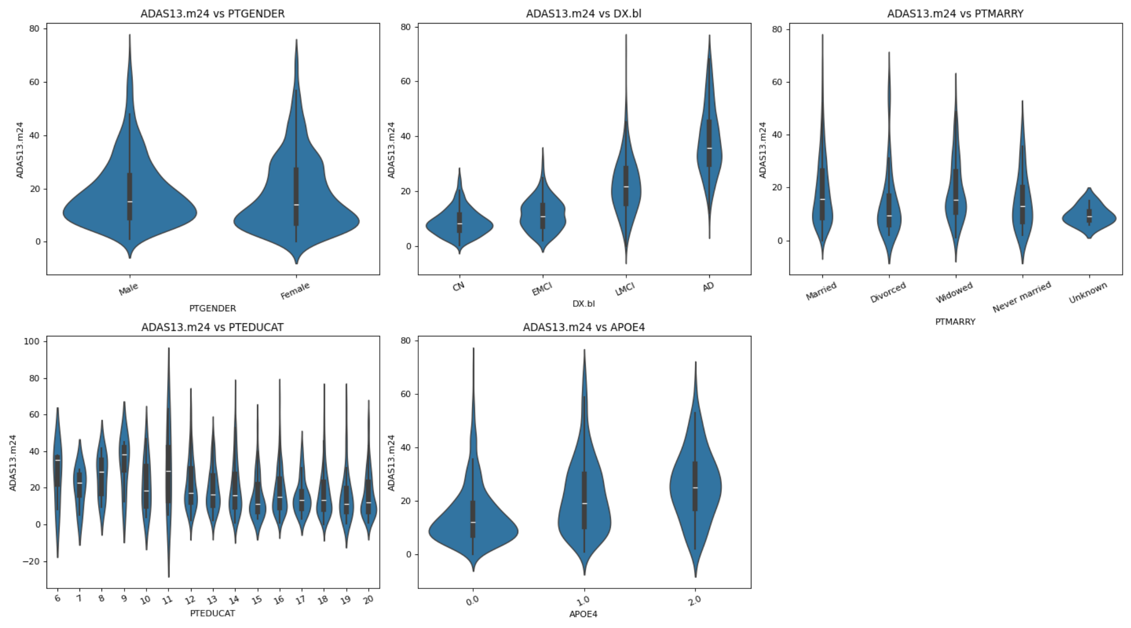


There is strong postive linear correlation between **ADAS13.bl** and the target, **ADAS13.m24**. This is to be expected and they refer to scores from the same test taken 24 months apart. There also appears to be a significantly weaker negative linear correlation between the target and the variables **Hippocampus**, **Entorhinal**, **Fusiform** and **Midtemp**.

We should note that there is a strong positive linear relationship between the variable **WholeBrain** and the variables **ICV**, **MidTemp** and **Fusiform** as well as between the variables **Fusiform** and **MidTemp** and the variables **Hippocampus** and **Entorhinal**. These strong linear relationships imply collinearity which can make it difficult to analyse the individual impacts of the variables on the target variable in regression analysis.

Below, we show violin plots of the target, **ADAS13.m24**, against the ordinal and categorical features including **PTGENDER**, **DX.bl**, **PTMARRY**, **PTEDUCAT**, and **APOE4**.

[212]:
```
# Violin plots of ordinal and categorical vs the target.
ordinal_categorical_columns = ['PTGENDER','DX.bl','PTMARRY','PTEDUCAT','APOE4']

# Define the desired order for DX.bl
dx_order = ['CN', 'EMCI', 'LMCI', 'AD']

fig, axes = plt.subplots(2, 3, figsize=(6 * 3, 10))
axes = axes.flatten()  # Flatten in case of a single row

for i, col in enumerate(ordinal_categorical_columns):
  if col == 'DX.bl':
    sns.violinplot(x=d['DX.bl'], y=d["ADAS13.m24"], order=dx_order, ax=axes[i])
  else:
    sns.violinplot(x=d[col], y=d["ADAS13.m24"], ax=axes[i])
  axes[i].set_title(f"ADAS13.m24 vs {col}")
  axes[i].tick_params(axis='x', rotation=25)  # Rotate x-axis labels

# Hide unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```



From these graphs the variables **APOE4** and **DX.bl** appear to be correlated to the **ADAS13.m24** score. This again makes sense as **DX.bl** is simply a categorical representation of each person's **ADAS13.bl** score, which we have seen from the scatterplots appears to be strongly positively correlated with the **ADAS13.m24** score. Furthermore, the e4 allele is medically known to increase susceptibility to Alzheimer's Disease and so we would expect to see a correlation between the number of the genes and the target score (Kim et al.). None of the other categorical variables present any strong evidence of correlation between themselves and the target variable.

3.5 Encoding Categorical and Ordinal Variables

The features **PTGENDER** and **PTMARRY** are inherently categorical, with no strict ordering, so we have encoded these features using one-hot encoding. Additionally, the feature **PTEDUCAT**, representing the number of years of education, is inherently ordinal. The violin plot shows a relatively smooth and consistent relationship between **PTEDUCAT** and the target, with no apparent thresholds or abrupt changes in the trend (ignoring some noise in the earlier categories with less individuals).

We chose to encode **DX.bl** using one-hot encoding. We considered encoding the feature as ordinal, as the diagnoses are ordered in increasing levels of severity. However, differences in the average target values were larger for higher levels of severity, indicating that the relationship was non-linear. This would have required the introduction of polynomial terms, decreasing the interpretability of the model.

In contrast, we chose to treat **APOE4** as an ordinal variable, rather than encoding as a categorical. This approach preserves the natural ordering, which aligns with the monotonic increase in average target values as **APOE4** increases. The relationship also appears linear, thus not requiring polynomial terms.

[213]:
```
# columns used in model fitting section
ordinal_columns = ['PTEDUCAT','APOE4']
categorical_columns = ['PTGENDER','PTMARRY','DX.bl']
log_numerical_columns = ["ADAS13.bl", "Ventricles"]
numerical_columns = ["AGE", "Hippocampus", "WholeBrain", "Entorhinal",␣
 ↪"Fusiform", "MidTemp",'ICV']



num_pre = Pipeline([
    ("num_scale", StandardScaler())
])

cat_pre = Pipeline([
    ('cat_encode', OneHotEncoder(drop='first', handle_unknown="ignore")) #␣
 ↪handle unknown tells it how to handle categories that weren't seen during␣
 ↪training
])

ord_poly_pipeline = Pipeline([
    ('ord_encode', OrdinalEncoder(categories=[dx_order]))
])

def log1p(x):
    return np.log(1 + x)
```

```
log_pre = FunctionTransformer(log1p, feature_names_out=lambda self, names:␣
 ↪names)


pre_proc = ColumnTransformer([
            ('log_pre', log_pre, log_numerical_columns),
            ('cat_pre', cat_pre, categorical_columns),
            ('num_pre', num_pre, numerical_columns),
            ('ord_pre', "passthrough", ordinal_columns)
        ])
```

[214]:
```
# drop log transformed variables used for visualising, these will be␣
 ↪reintroduced later in the pipeline
d = d.drop(['lADAS13.bl','lVentricles', 'ADAS13.m24'], axis=1)

d = d.dropna()

X = d.drop('lADAS13.m24', axis = 1)
y = d['lADAS13.m24']

# Create train and test data frames
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
 ↪random_state = 42)
```

4 Model Fitting and Tuning

We make use of several helper functions in our model analysis; these are contained in the python cell below.

[215]:
```
def get_coefs(m):
    """Returns the model coefficients from a Scikit-learn model object as an␣
 ↪array,
    includes the intercept if available.
    """

    # If pipeline, use the last step as the model
    if (isinstance(m, sklearn.pipeline.Pipeline)):
        m = m.steps[-1][1]


    if m.intercept_ is None:
        return m.coef_

    return np.concatenate([[m.intercept_], m.coef_])

def model_fit(m, X, y, plot = False):
    """Returns the mean squared error, root mean squared error and R^2 value of␣
↪a fitted model based
    on provided X and y values.

    Args:
        m: sklearn model object
        X: model matrix to use for prediction
        y: outcome vector to use to calculating rmse and residuals
        plot: boolean value, should fit plots be shown
    """

    y_hat = m.predict(X)
    MSE = mean_squared_error(y, y_hat)
    RMSE = np.sqrt(mean_squared_error(y, y_hat))
    Rsqr = r2_score(y, y_hat)

    Metrics = (round(MSE, 4), round(RMSE, 4), round(Rsqr, 4))

    res = pd.DataFrame(
        data = {'y': y, 'y_hat': y_hat, 'resid': y - y_hat}
    )

    if plot:
        plt.figure(figsize=(12, 6))

        plt.subplot(121)
        sns.lineplot(x='y', y='y_hat', color="grey", data =  pd.
↪DataFrame(data={'y': [min(y),max(y)], 'y_hat': [min(y),max(y)]}))
        sns.scatterplot(x='y', y='y_hat', data=res).set_title("Observed vs␣
↪Fitted values")

        plt.subplot(122)
        sns.scatterplot(x='y_hat', y='resid', data=res).set_title("Fitted␣
↪values vs Residuals")
        plt.hlines(y=0, xmin=np.min(y), xmax=np.max(y), linestyles='dashed',␣
↪alpha=0.3, colors="black")

        plt.subplots_adjust(left=0.0)

        plt.suptitle("Model (MSE, RMSE, Rsq) = " + str(Metrics), fontsize=14)
        plt.show()

    return MSE, RMSE, Rsqr
```

We began our analysis examining initial naive models, which could adequately explain variance but were grossly dependent on a large number of low-magnitude coefficients. Implementing polynomial features produced a model with greater and more unexplained variance; linear regression was a clear superior fit. In each evolution of the model, we fitted both the data sets with missing data imputed and those with missing data dropped; there was no significant disruption to fit in either case.

We attempted to use ridge regression to penalize overfitting the numerous variables, but found that the model affected all of the small variables too similarly to differentiate. We hence looked towards lasso regression to tune our model further.

To find an optimal hyperparameter α in the lasso regression, we implemented a grid search cross validation, as held in the code cell below:

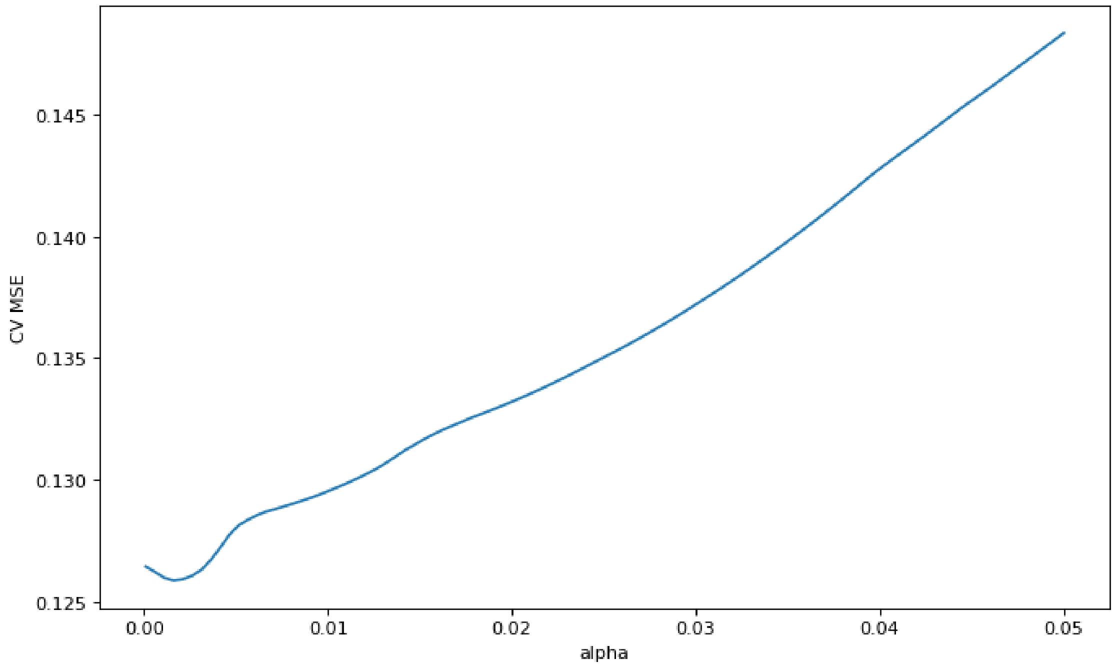
[240]:
```
alphas = np.linspace(0.0001, 0.05, num=100)

cv = KFold(5, shuffle=True, random_state=1)


lass_gs = Pipeline([
    ("pre_processing", pre_proc),
    ('model', Lasso())
])

gs_L = GridSearchCV(lass_gs,
    param_grid={'model__alpha': alphas},
    cv=cv,
    scoring="neg_mean_squared_error")


gs_L.fit(X_train, y_train)
```

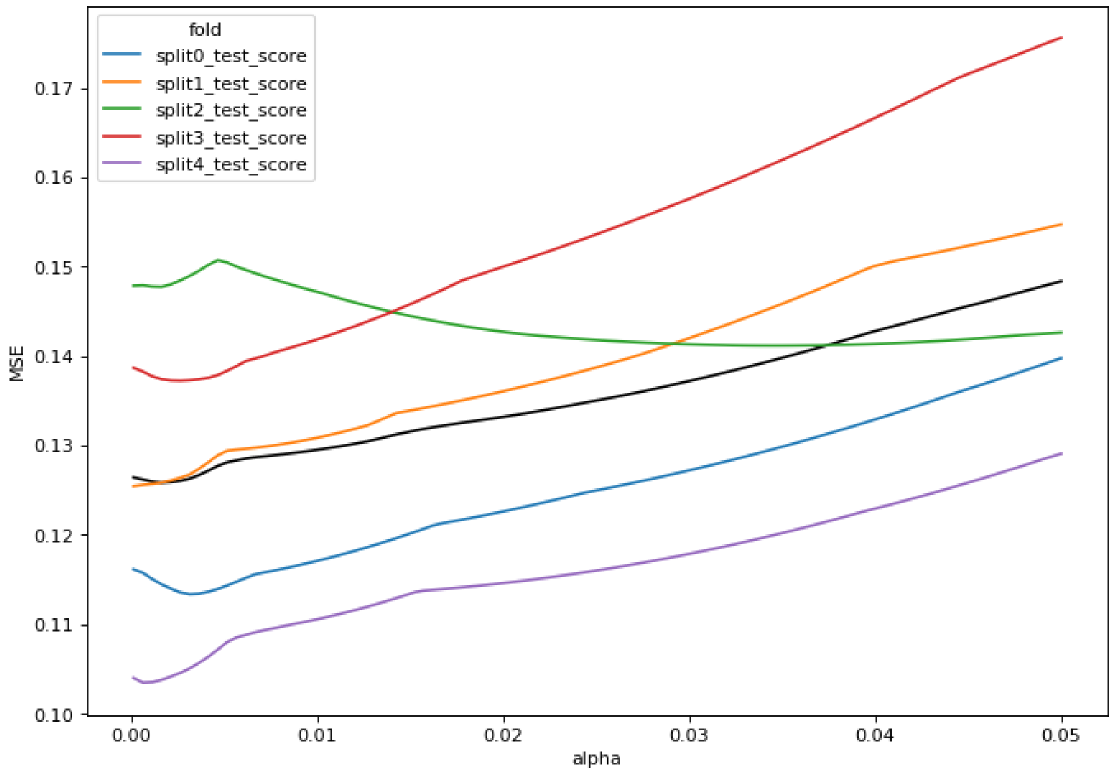
[240]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True),
             estimator=Pipeline(steps=[('pre_processing',
ColumnTransformer(transformers=[('log_pre',
FunctionTransformer(feature_names_out=<function <lambda> at 0x17af00cc0>,
             func=<function log1p at 0x17af00040>),
['ADAS13.bl',
'Ventricles']),
('cat_pre',
Pipeline(steps=[('cat_encode',
          OneHotEncoder(drop='first',…
       0.03538283, 0.03588687, 0.03639091, 0.03689495, 0.03739899,
       0.03790303, 0.03840707, 0.03891111, 0.03941515, 0.03991919,
       0.04042323, 0.04092727, 0.04143131, 0.04193535, 0.04243939,
       0.04294343, 0.04344747, 0.04395152, 0.04445556, 0.0449596 ,
       0.04546364, 0.04596768, 0.04647172, 0.04697576, 0.0474798 ,
       0.04798384, 0.04848788, 0.04899192, 0.04949596, 0.05      ])},
             scoring='neg_mean_squared_error')
```

In plotting cross validated mean square error against values of α, we found a clear trough about $\alpha \approx 0.002$; this agrees with the `best_params_` feature in `sklearn`. In examining particular folds of the cross validation under different random seeds, we found clear consistency in functions of mean squared error against α; see these plots below.

```
[241]: cv_mse = pd.DataFrame(
           data = gs_L.cv_results_
       ).filter(
           # Extract the split#_test_score and mean_test_score columns
           regex = '(split[0-9]+|mean)_test_score'
       ).assign(
           # Add the alphas as a column
           alpha = alphas
       )

       cv_mse.update(
           # Convert negative mses to positive
           -1 * cv_mse.filter(regex = '_test_score')
       )

       plt.figure(figsize=(10,6))
       ax = sns.lineplot(x='alpha', y='mean_test_score', data=cv_mse)
       ax.set_ylabel('CV MSE')
       plt.show()

       cv_mse = pd.DataFrame(
           data = gs_L.cv_results_
       ).filter(
           # Extract the split#_test_score and mean_test_score columns
           regex = '(split[0-9]+|mean)_test_score'
       ).assign(
           # Add the alphas as a column
           alpha = alphas
       )

       cv_mse.update(
           # Convert negative mses to positive
           -1 * cv_mse.filter(regex = '_test_score')
       )

       # Reshape the data frame for plotting
       d = cv_mse.melt(
           id_vars=('alpha','mean_test_score'),
           var_name='fold',
           value_name='MSE'
       )

       # Plot the validation scores across folds
       plt.figure(figsize=(10,7))
```

```
sns.lineplot(x='alpha', y='MSE', color='black', errorbar=None, data = d)  #␣
↪Plot the mean MSE in black.
sns.lineplot(x='alpha', y='MSE', hue='fold', data = d) # Plot the curves for␣
↪each fold in different colors
plt.show()

alph = gs_L.best_params_
print(f'The best cross validated hyper parameter was alpha={alph}')
```





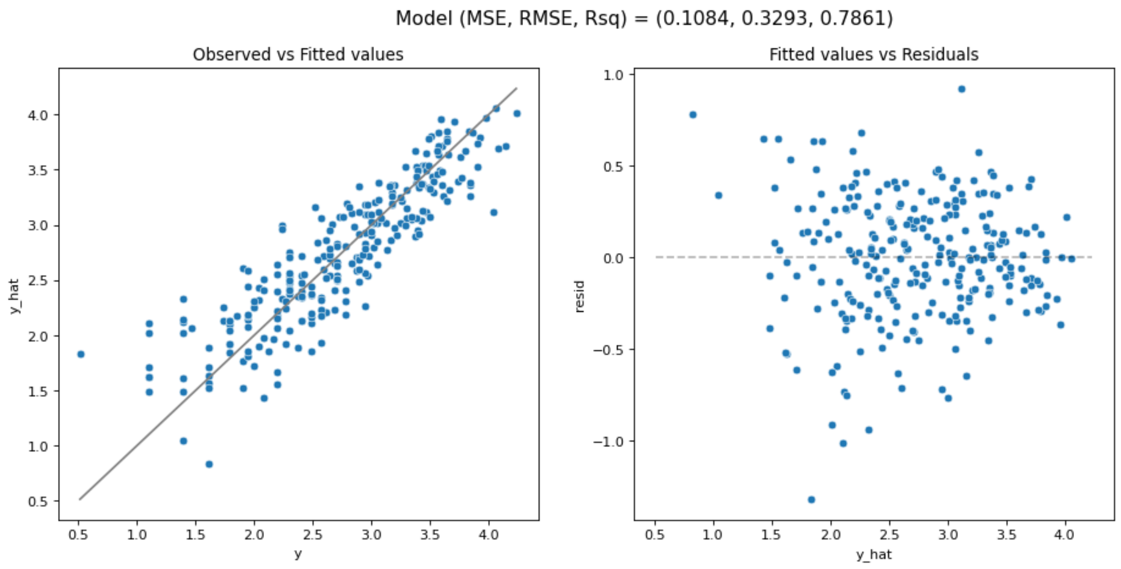
```
The best cross validated hyper parameter was alpha={'model__alpha':
0.0016121212121212123}
```

For this value of α, marital status, ventricle volume, and whole brain volume were deemed unimportant to prediciton. Our final model used lasso regression on this reduced dataset with the same feature engineering.

Performance of this model against training data is detailed in the code cell below; the high R^2 value, low mean squared error, and homoscedasticity of residuals, especially for greater fitted values, indicate a promising fit.

[245]:
```
X_train_fin = X_train.drop(['PTMARRY', 'Ventricles', 'WholeBrain'], axis=1)

X_test_fin = X_test.drop(['PTMARRY', 'Ventricles', 'WholeBrain'], axis=1)




categorical_columns_fin = ['PTGENDER', 'DX.bl']
log_numerical_columns_fin = ["ADAS13.bl"]
numerical_columns_fin = ["AGE", "Hippocampus", "Entorhinal", "Fusiform",␣
↪"MidTemp",'ICV']
ordinal_columns_fin = ['PTEDUCAT','APOE4']
```

```
pre_proc_fin = ColumnTransformer([
            ('log_pre', log_pre, log_numerical_columns_fin),
            ('cat_pre', cat_pre, categorical_columns_fin),
            ('num_pre', num_pre, numerical_columns_fin),
            ('ord_pre', "passthrough", ordinal_columns_fin)
        ])

reg_pipe_base_fin= Pipeline([
    ("pre_processing", pre_proc_fin),
    ('model', Lasso(alpha=alph['model__alpha']))
])

final_model = reg_pipe_base_fin.fit(X_train_fin, y_train)
model_fit(final_model, X_test_fin, y_test, plot=True)
```



```
[245]: (0.10843448469881958, 0.3292939184054567, 0.7860767017776813)
```

This model has non-zero coefficients as listed under the code cell below.

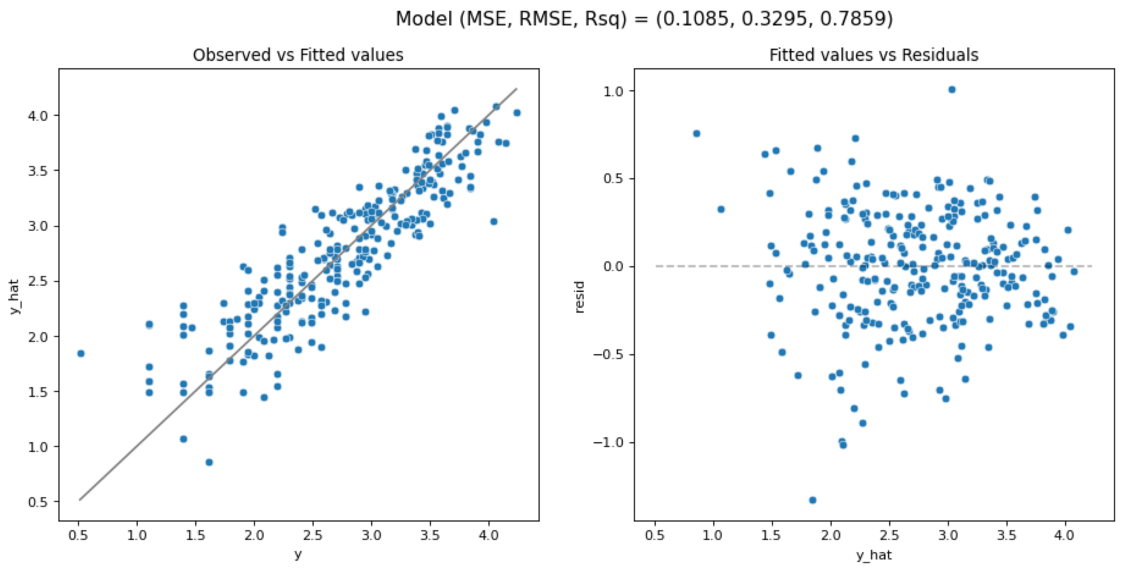
```
[242]: coefs = get_coefs(final_model)
par_names = final_model["pre_processing"].get_feature_names_out()
print(f'intercept : {coefs[0]}')
for i in range(len(coefs) - 1):
    print(f"{par_names[i]} : {coefs[i + 1]}")
```

```
intercept : 0.9540090616926329
log_pre__ADAS13.bl : 0.7919656863372729
```

```
cat_pre__PTGENDER_Male : 0.023543617453704143
cat_pre__DX.bl_CN : -0.37094826032566336
cat_pre__DX.bl_EMCI : -0.28280492175766997
cat_pre__DX.bl_LMCI : -0.13269817532702313
num_pre__AGE : -0.015373381090306781
num_pre__Hippocampus : -0.038593065913878406
num_pre__Entorhinal : -0.006734885223762129
num_pre__Fusiform : -0.02816027360099717
num_pre__MidTemp : -0.07624015148243958
num_pre__ICV : 0.06336010954809912
ord_pre__PTEDUCAT : -0.010099773241111354
ord_pre__APOE4 : 0.047579040016 11129
```

The figures below show the same plots as fitted for our baseline model and our final tuned model for comparison; applying both models to our test data, we plot observed against fitted values and fitted values against residuals.

[243]:
```python
# baseline model with fit and residuals

reg_pipe_base= Pipeline([
    ("pre_processing", pre_proc),
    ('model', LinearRegression())
])

fitted_model = reg_pipe_base.fit(X_train, y_train)

model_fit(fitted_model, X_test, y_test, plot=True)
```



[243]:
```
(0.1085375255259302, 0.3294503384820392, 0.7858734192735196)
```

5 Discussion & Conclusions

Race and ethnicity are known to influence disease risk in many cases, which may include Alzheimer's, as certain populations may be more susceptible than others. Thus, we should be wary of removing such features from any model monitoring disease progression. However, in this dataset, the features **PTRACCAT** and **PTETHCAT** are highly imbalanced, with only 27 individuals outside the Not Hispanic/Latino group and 65 individuals outside the White category. This creates two key issues: the model will primarily fit to the majority group, and in addition, any effect observed in the minority categories may be biased and unreliable due to the small sample size.

While we could apply techniques such as oversampling minority groups or undersampling the majority class, these methods risk either introducing synthetic patterns that do not generalise, or discarding valuable data. Additionally, the violin plots as detailed in the exploratory data analysis suggest no clear relationship between these features and the target variable, **ADAS13.m24**. Given the limited representation of minority groups, any patterns learned would likely be misleading rather than informative. Therefore, we exclude these two features to prevent the model from fitting spurious associations and avoid drawing unreliable conclusions. With this in mind, applying the results of our model to individuals in these racial/ethnic groups has not been tested, and could be highly unreliable. It may be an avenue of future study to collect data with a larger sample of minority racial/ethnic groups to better understand differences in risk factors between racial/ethnic groups.

Based on the coefficients of our final model, the most significant factor for risk of higher cognitive decline is the original baseline ADAS13 score. The diagnosis of the individual at the baseline visit, **DX.bl**, also has a weak relationship with their predicted ADAS13 score after 24 months. Some other factors that may indicate a risk of cognitive decline are biological factors, including greater intracranial volume and lesser middle temporal lobe volume, which appear associated with an increased risk of cognitive decline. These relationships were not found to be extremely strong, and very much subsidiary to the patient's baseline ADAS score. However these biological factors may be useful in indicating useful drugs/therapies in treating dementia and pointing towards directions of research since they refer to specific brain regions.

Our final model has better performance at predicting higher levels of cognitive decline than lower levels. This is a desirable feature, as it means that when using it to assess a patient, it is less likely that their high risk of cognitive decline will be underestimated, whereas lower precision of estimates for lower levels of cognitive decline will at worst cause more caution to be taken with a patient than is necessary.

It should be noted that our final model produces an only marginally better fit to the data and our assumptions than our baseline. However, as this model is based on a reduced data set, it is useful in highlighting particular parameters that are more important to drawing conclusions, and serves this end over producing better predictive performance.

With a high test R^2 of 0.785, our model is able to explain a significant proportion of the variability in the target. However, it should be kept in mind that that due to the high variance in the data and low overall relationship between all features other than the individuals baseline score, there is still a significant amount of variability left unexplained.

Since the feature with by far the strongest relationship to the follow-up cognitive score is the baseline cognitive score, we recommend further study tracks the cognitive score more regularly, rather than only once every 24 months. Further study of the progression may lead to further insight into effective prediction.

6 Generative AI statement

We used Generative AI to understand `sklearn` Pipeline functionalities, but not to create code or prose. We listened to AI Generated music during the production of this report.

7 References

Kim J, Basak JM, Holtzman DM. The role of apolipoprotein E in Alzheimer's disease. Neuron. 2009 Aug 13;63(3):287-303. doi: 10.1016/j.neuron.2009.06.026. PMID: 19679070; PMCID: PMC3044446.

```
[2]: # Run the following to render to PDF
     !jupyter nbconvert --to pdf Project1FINAL.ipynb
```

```
[NbConvertApp] Converting notebook Project1FINAL.ipynb to pdf
[NbConvertApp] Support files will be in Project1FINAL_files\
[NbConvertApp] Making directory .\Project1FINAL_files
[NbConvertApp] Writing 87767 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | b had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 1506795 bytes to Project1FINAL.pdf
```