In [59]:
```
'''
This jupyter notebook includes code taken from the following nilearn project,
"8.3.10. Voxel-Based Morphometry on Oasis dataset", which predicts age from
grey matter morhpometry. Feature redux = k-best ANOVA, prediction function = SVM

I am tweaking the code to the following analysis:
Feature redux = PCA, prediction function = SVM, then random forest.
Including various plots.
'''
```

In [ ]:
```
# Let's keep our notebook clean, so it's a little more readable!
import warnings
warnings.filterwarnings('ignore')
```

In [51]:
```
from nilearn import datasets
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
from nilearn.input_data import NiftiMasker
```

In [60]:
```
from nilearn import plotting
```

In [53]:
```
oasis_dataset = datasets.fetch_oasis_vbm()
gray_matter_map_filenames = oasis_dataset.gray_matter_maps
age = oasis_dataset.ext_vars['age'].astype(float, None)
```

In [74]:
```
#Checking to see where the dataset is downloaded onto my computer
oasis_dataset
```

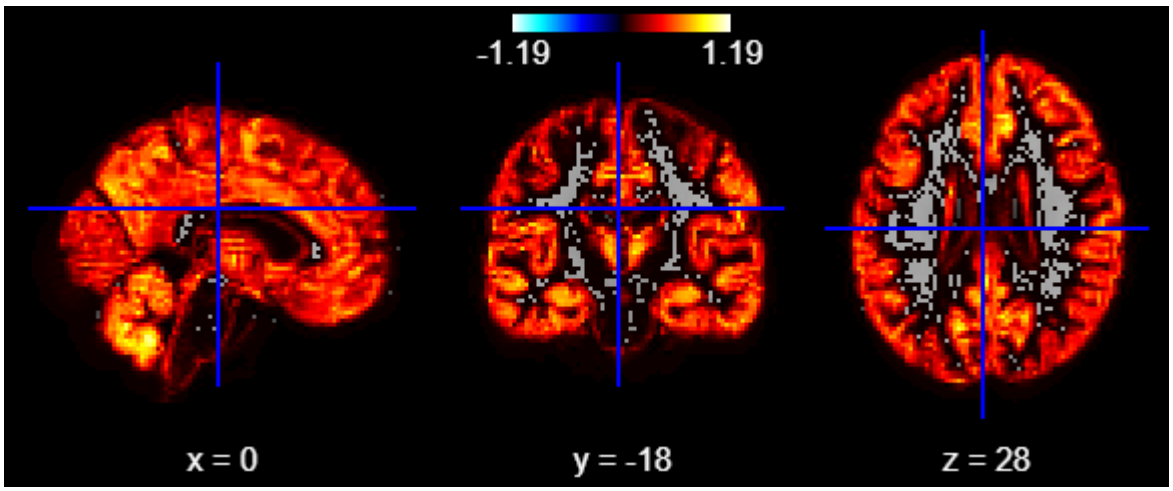
Out[74]:
```
{'gray_matter_maps': ['C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0001_MR1
\\mwrc1OAS1_0001_MR1_mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0002_MR1\\mwrc1OAS1_0002_MR1_
mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0003_MR1\\mwrc1OAS1_0003_MR1_
mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0004_MR1\\mwrc1OAS1_0004_MR1_
mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0005_MR1\\mwrc1OAS1_0005_MR1_
mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0006_MR1\\mwrc1OAS1_0006_MR1_
mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0007_MR1\\mwrc1OAS1_0007_MR1_
mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0009_MR1\\mwrc1OAS1_0009_MR1_
mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0010_MR1\\mwrc1OAS1_0010_MR1_
mpr_anon_fslswapdim_bet.nii.gz',
  'C:\\Users\\rwick/nilearn_data\\oasis1\\OAS1_0011_MR1\\mwrc1OAS1_0011_MR1_
```

In [55]:
```
'''
#for later if I decide to look at white matter
#Note to self: make sure the nifti masker is adapted for white matter, with corre
white_matter_map_filenames = oasis_dataset.white_matter_maps
wm_maps_masked = nifti_masker.fit_transform(white_matter_map_filenames)
'''
```

In [ ]:
```
nifti_masker = NiftiMasker(
    standardize=False,
    smoothing_fwhm=2,
    memory='nilearn_cache')  # cache options
gm_maps_masked = nifti_masker.fit_transform(gray_matter_map_filenames)

n_samples, n_features = gm_maps_masked.shape
print("%d samples, %d features" % (n_subjects, n_features))
```

In [61]:
```
subject1 = oasis_dataset.gray_matter_maps[0]
plotting.view_img(subject1)
```

Out[61]:

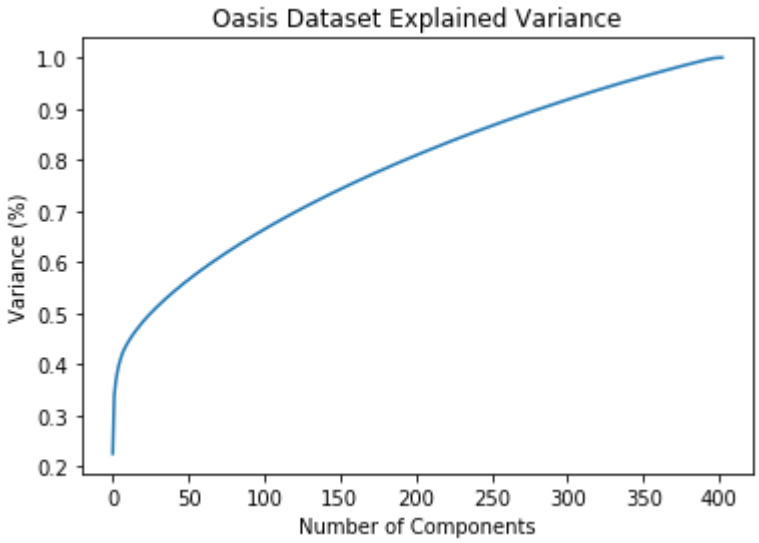


In [58]:
```
print("ANOVA + SVR")
# Define the prediction function to be used.
# Here we use a Support Vector Classification, with a linear kernel
from sklearn.svm import SVR
svr = SVR(kernel='linear')

# Dimension reduction
from sklearn.feature_selection import VarianceThreshold, SelectKBest, \
        f_regression

# Remove features with too low between-subject variance
variance_threshold = VarianceThreshold(threshold=.01)

# Here we use a classical univariate feature selection based on F-test,
# namely Anova.
feature_selection = SelectKBest(f_regression, k=2000)

# We have our predictor (SVR), our feature selection (SelectKBest), and now,
# we can plug them together in a *pipeline* that performs the two operations
# successively:
from sklearn.pipeline import Pipeline
anova_svr = Pipeline([
            ('variance_threshold', variance_threshold),
            ('anova', feature_selection),
            ('svr', svr)])

### Fit and predict
anova_svr.fit(gm_maps_masked, age)
age_pred = anova_svr.predict(gm_maps_masked)
```

```
ANOVA + SVR
```

In [68]:
```
print("PCA + SVR")
# Define the prediction function to be used.
# Here we use a Support Vector Classification, with a linear kernel
from sklearn.svm import SVR
svr = SVR(kernel='linear')

# Dimension reduction

from sklearn.decomposition import PCA
#from sklearn.preprocessing import MinMaxScaler -
#not needed because values are between 0 and 1, as they are tissue probabilities
feature_selection = PCA() # Here I'm not setting any hyperparameters on the numbe
feature_selection.fit(gm_maps_masked) # Fit the data.

print("Plotting the Cumulative Summation of the Explained Variance")
plt.figure()
plt.plot(np.cumsum(feature_selection.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Variance (%)') #for each component
plt.title('Oasis Grey Matter Explained Variance')
plt.show()

print("Plotting a scree plot with PCA components")
#Code here

#Include a correlation matrix between all of my components. What would I expect t

'''
We have our predictor (SVR), our feature selection (PCA), and now,
we can plug them together in a *pipeline* that performs the two operations
successively:
'''

#Adapt this pipeline for PCA

from sklearn.pipeline import Pipeline
pca_svr = Pipeline([
            #'variance_threshold', variance_threshold), #not sure what to specify
            ('PCA', feature_selection),
            ('svr', svr)])

# Fit and predict
pca_svr.fit(gm_maps_masked, age)
age_pred = anova_svr.predict(gm_maps_masked)
```

```
PCA + SVR
```


Oasis Dataset Explained Variance
Variance (%)
Number of Components
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
50
100
150
200
250
300
350
400

In [ ]:
```
coef = svr.coef_
# reverse feature selection
coef = feature_selection.inverse_transform(coef)
# reverse variance threshold
coef = variance_threshold.inverse_transform(coef)
# reverse masking
weight_img = nifti_masker.inverse_transform(coef)

# Create the figure
from nilearn.plotting import plot_stat_map, show
bg_filename = gray_matter_map_filenames[0]
z_slice = 0


fig = plt.figure(figsize=(5.5, 7.5), facecolor='k')
# Hard setting vmax to highlight weights more
display = plot_stat_map(weight_img, bg_img=bg_filename,
                        display_mode='z', cut_coords=[z_slice],
                        figure=fig, vmax=1)
display.title('SVM weights', y=1.2)

# Measure accuracy with cross validation
from sklearn.model_selection import cross_val_score
cv_scores = cross_val_score(anova_svr, gm_maps_masked, age)

# Return the corresponding mean prediction accuracy
prediction_accuracy = np.mean(cv_scores)
print("=== ANOVA ===")
print("Prediction accuracy: %f" % prediction_accuracy)
print("")

### Inference with massively univariate model ##################################
print("Massively univariate model")

# Statistical inference
from nilearn.mass_univariate import permuted_ols
data = variance_threshold.fit_transform(gm_maps_masked)
neg_log_pvals, t_scores_original_data, _ = permuted_ols(
    age, data,  # + intercept as a covariate by default
    n_perm=2000,  # 1,000 in the interest of time; 10000 would be better
    n_jobs=1)  # can be changed to use more CPUs
signed_neg_log_pvals = neg_log_pvals * np.sign(t_scores_original_data)
signed_neg_log_pvals_unmasked = nifti_masker.inverse_transform(
    variance_threshold.inverse_transform(signed_neg_log_pvals))

# Show results
threshold = -np.log10(0.1)  # 10% corrected

fig = plt.figure(figsize=(5.5, 7.5), facecolor='k')

display = plot_stat_map(signed_neg_log_pvals_unmasked, bg_img=bg_filename,
                        threshold=threshold, cmap=plt.cm.RdBu_r,
                        display_mode='z', cut_coords=[z_slice],
                        figure=fig)
title = ('Negative $\log_{10}$ p-values'
         '\n(Non-parametric + max-type correction)')
```

```
display.title(title, y=1.2)

n_detections = (signed_neg_log_pvals_unmasked.get_data() > threshold).sum()
print('\n%d detections' % n_detections)

show()
```

In [69]:
```
#All k-fold cross-validation stuff here
from sklearn.model_selection import train_test_split
```