Online Supplemental Materials

Section 1: Preliminary Analyses Examining Relationships Between Demographic Variables and Academic Outcomes and Possible Identity Scores

Table S1.

Correlation between child and school level measures

Measure	1	2	3	4	5	6	7	8
1. FRPL (School-Level)	--							
2. FRPL (Child-level)	0.188**	--						
3. Latinx (School-Level)	0.072	-0.051	--					
4. Latinx (Child-level)	0.098	-0.009	0.763**	--				
5. Black (School-Level)	0.020	0.070	-0.995**	-0.758**	--			
6. Black (Child-level)	0.002	0.074	-0.824**	-0.898**	0.829**	--		
7. Female (Student-Level)	-0.043	0.045	-0.044	-0.084	0.042	0.110	--	
8. Student-Teacher Ratio	-0.063	-0.018	0.410**	0.268**	-0.378**	-0.300**	0.008	--

Note: ** p<.01; FRPL = Free or reduced-price lunch status

Table S2.

Correlation between child and school level measures and GPA

	FRPL (School-Level)	FRPL (Child-level)	Latinx (School-Level)	Latinx (Child-level)	Female (Student-Level)	Student-Teacher Ratio
Core GPA 6th	-0.263**	-0.143*	-0.076	-0.073	0.280**	0.182**
Core GPA 7th	-0.095	-0.111	-0.176**	-0.141*	0.284**	-0.046
Core GPA 8th	-0.224**	-0.155*	-0.050	-0.053	0.299**	0.089
6th-7th GPA Change	0.280**	0.050	-0.174**	-0.118	0.015	-0.386**
7th-8th GPA Change	-0.165**	-0.029	0.255**	0.183**	-0.075	0.223**

Note: * p<.01, ** p<.01; FRPL = Free or reduced-price lunch status

Table S3.

Correlation between child and school level measures and possible identity scores

	FRPL (School-Level)	FRPL (Child-level)	Latinx (School-Level)	Latinx (Child-level)	Female (Student-Level)	Student-Teacher Ratio
Fall School-Focused PI Count	0.111	0.047	-0.024	-0.009	-0.032	0.011
Fall School-Focused PI Balance Count	0.058	0.038	0.008	0.037	-0.003	0.022
Fall School-Focused PI with Strategies Count	0.081	0.021	-0.031	-0.048	-0.064	0.001
Fall School-Focused PI Plausibility	0.064	0.085	-0.031	-0.038	0.029	0.076
Spring School-Focused PI Count	0.081	-0.032	0.061	0.120	0.093	-0.082
Spring School-Focused PI Balance Count	0.103	0.062	0.048	0.089	0.128*	-0.008
Spring School-Focused PI with Strategies Count	0.053	-0.025	-0.019	0.050	0.067	-0.116
Spring School-Focused PI Plausibility	0.083	-0.011	0.048	0.099	0.152*	0.006

School-Focused PI Count Change	0.059	-0.043	0.068	0.125*	0.102	-0.087
School-Focused PI Balance Count Change	0.092	0.054	0.047	0.083	0.133*	-0.014
School-Focused PI with Strategies Count Change	0.037	-0.03	-0.013	0.062	0.082	-0.119
School-Focused PI Plausibility Change	0.072	-0.029	0.055	0.109	0.149*	-0.009

Note: PI=possible identity

Table S4 Post-hoc sensitivity analyses for Models 1, 3, 4 (Table 5) with *n*=247: Possible Identity Change Scores Predicting End-of-Year GPA

Model 1 -- change score predicting end-of-8th-grade GPA

School-focused Identities	Model 1		Model 3		Model 4	
	Effect Size (f_2)	Power	Effect Size (f_2)	Power	Effect Size (f_2)	Power
Count	.037	.86	.039	.87	.039	.87
Balance	.033	.81	.037	.86	.036	.85
Count + Strategies	.035	.83	.042	.89	.041	.89
Plausibility	.051	.94	.030	.78	.028	.74

Note: Model 1 = no controls, Model 3 = controlling for school variables and prior GPA, Model 4 = controlling for school variables, prior GPA, and demographics

Table S5.

Effect of interaction between school focused possible identity scores and school-level context variables on 8th grade core GPA

	School-Level FRPL			School Level Latinx			Student/Teacher Ratio		
Predictor	B	*95% CI*	*p*	B	*95% CI*	*p*	B	*95% CI*	*p*

School-focused possible identity count	.005	-.013, .022	.595	.001	-.003, .005	.626	.040	-.116, .036	.305
School-focused possible identity balance count	.000	-.018, .019	.979	-.001	-.006, .003	.634	-.037	-.111, .037	.324
School-focused possible identities with strategies count	.010	-.007, .027	.263	.002	-.002, .007	.300	-.011	-.085, .063	.774
School-focused possible identities plausibility score	.013	-.004, .029	.131	.003	-.002, .007	.221	.017	-.056, .089	.654

Notes: In each model the dependent variable is 8th grade core grade point average; the regression coefficients in each cell are for the interaction between the possible identity metric in the left column and the school-level context variables in the top row.

Section 2: File Setup and Python Code for Developing and Using Possible Selves Classifier

File Setup

The code is written to read possible self training data in the following format (The code ensures that only the possible self code is used to train the algorithm.)

ID	Possible Self	CODE	Strategy	CODE
1	On honor roll	1	Study every day	1
2	More popular	2	Hang out after school	2
3	Playing my x-box	5	Do my chores	5

The code is written to classify or code uncoded possible self data in the following format:

ID	Possible Self	Strategy
1	On honor roll	Study every day
2	More popular	Hang out after school
3	Playing my x-box	Do my chores

Running the code will output possible self classification in the format below. The 'Combined" column shows which words in the responses were used by the algorithm. The 'Vecs' column can be ignored. The 'Class' column has the possible self code generated by the algorithm.

ID	Possible Self	Strategy	Combined	Vecs	Class
1	On honor roll	Study every day	['honor', 'roll', 'study', 'every', 'day']	[-0.0802 -0.34582]	1
2	More popular	Hang out after school	['More', 'pouplar', 'Hang', 'after', 'school']	[1.416e-01 5.060e-01]	2
3	Playing my x-box	Do my chores	['Playing', 'x-box', 'chores']	[-0.02734 0.103027]	0

Python Code

```
#Import the necessary packages

import pandas as pd
import gensim
from autocorrect import spell
import string
from nltk.corpus import stopwords
import numpy as np
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.model_selection import ShuffleSplit
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
import xlrd
import openpyxl

#Defining classification functions

def CombineFiles(files):
   t = pd.DataFrame()
   list = []
   for file in files :
      data = pd.read_excel(file)
      list.append(data)
   return pd.concat(list,ignore_index=True)

def listCreation(X):
   X_n = []
```

```
    for i in X:
        X_n.append(i)
    return X_n

def combineClass(class_codes,t):
    df = t.copy()
    for i in class_codes:
        df.ix[df.CODE == i, 'CODE'] = 0
    return df

def classifyingResult(cv,classifier,X,Y):
    accuracies = []
    for train_index, test_index in cv.split(X):
        X_tr = [X[i - 1] for i in train_index]
        X_tes = [X[i - 1] for i in test_index]
        y_tr = [Y[i - 1] for i in train_index]
        y_tes = [Y[i - 1] for i in test_index]
        clf = classifier.fit(X_tr, y_tr)
        y_pred = clf.predict(X_tes)
        a = accuracy_score(y_tes, y_pred)
        c =confusion_matrix(y_tes, y_pred)
        accuracies.append(a)
        print a
        print c
    print "Mean Accuracy : " , np.mean(accuracies)


def classifyingResultWithSeprateTrainTest(X_tr,y_tr,X_tes):
    classifier_SVM = svm.SVC(kernel='linear',decision_function_shape='ovr')
```

```
    classifier = classifier_SVM
    clf = classifier.fit(X_tr, y_tr)
    y_pred = clf.predict(X_tes)
    t_test['Class']=y_pred
    t_test.to_excel('output_cycle.xlsx')

#Defining stopwords and importing word representations
#Google doc2vec file location can be downloaded from https://github.com/mmihaltz/word2vec-GoogleNews-vectors

stop = set(stopwords.words('english'))
model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)



#This section first combines files with training data (if there are multiple files) then preprocess them (including ensuring there is a single code -- the possible self code -- for each response), and combines necessary classes (here, 3, 4, 5, and 7) into one class

files = [[Names of Files with training data in them]]
t = CombineFiles(files)
t = t[pd.notnull(t['CODE'])]
t.loc[t['CODE'] != t['CODE.1'], 'Strategy'] = ''
t["Combined"] = t["Possible Self"] +' '+ t["Strategy"]
t['Combined'].fillna('', inplace=True)
t['Combined'] = t['Combined'].apply(lambda x:x.encode('utf-8'))
t['Combined'] = t['Combined'].apply(lambda x:str(x))
t['Combined'] = t['Combined'].apply(lambda x:x.lower())
t['Combined'] = t['Combined'].apply(lambda x:x.translate(None,string.punctuation))
t['Combined'] = t['Combined'].apply(lambda x: [spell(str(item)) for item in x.split() if
        spell(str(item)) not in stop and spell(str(item)) in model.vocab])
```

```
t['Vecs'] = t['Combined'].apply(lambda s:reduce(lambda x, y: x + y, map(lambda e:
        np.array(model[e]), s)) if len(s)!=0 else np.zeros(300, dtype='float32'))

original = t.copy()
class_codes= [3,4,5,7] #combining classes 3, 4, 5, 7 (or whichever classes you wish to combine into one class)
t = combineClass(class_codes,original)
X = t['Vecs']
Y = t['CODE']
X = listCreation(X)
Y = listCreation(Y)

print confusion_matrix(Y,Y) #To get an idea of the data distribution

#Split the data based on parameters provided
n_splits = 10
test_size = 0.15

#10-fold cross validation SVM classification
cv = ShuffleSplit(n_splits=n_splits, test_size=test_size, random_state=0)
classifier_SVM = svm.SVC(kernel='linear',decision_function_shape='ovr')
classifier = classifier_SVM
classifyingResult(cv,classifier,X,Y)


#this section of the code should be run if a seperate test file needs to be coded. First, it trains on the combined pre-processed files from the previous portion, then it preprocesses the test file, and runs the classification

files_test = [Name of file with data to be coded]
t_test = CombineFiles(files_test)
```

```
t_test['Strategy'].fillna('', inplace=True)
t_test['Possible Self'].fillna('', inplace=True)
t_test["Combined"] = t_test["Possible Self"] +' '+ t_test["Strategy"]
t_test['Combined'].fillna('', inplace=True)
t_test['Combined'] = t_test['Combined'].apply(lambda x:x.encode('utf-8'))
t_test['Combined'] = t_test['Combined'].apply(lambda x:str(x))
t_test['Combined'] = t_test['Combined'].apply(lambda x:x.lower())
t_test['Combined'] = t_test['Combined'].apply(lambda x:x.translate(None,string.punctuation))
t_test['Combined'] = t_test['Combined'].apply(lambda x: [spell(str(item)) for item in x.split() if
        spell(str(item)) not in stop and spell(str(item)) in model.vocab])
t_test['Vecs'] = t_test['Combined'].apply(lambda s:reduce(lambda x, y: x + y, map(lambda e:
        np.array(model[e]), s)) if len(s)!=0 else np.zeros(300, dtype='float32'))
original_test = t_test.copy()

X_test = t_test['Vecs']
X_test = listCreation(X_test)

classifyingResultWithSeprateTrainTest(X,Y,X_test)
```