A machine learning approach to predicting academic performance in Pennsylvania's schools

Shan Chen [1], Yuanzhao Ding [2,*]

[1] Department of Applied Social Sciences, The Hong Kong Polytechnic University, 11 Yuk Choi Rd, Hung Hom, Hong Kong, China; chenshan1893@gmail.com

[2] School of Geography and the Environment, University of Oxford, South Parks Road, Oxford OX1 3QY, UK

*Correspondence: armstrongding85@gmail.com

Table S1. Coding for machine learning and data analysis (Python)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

data = pd.read_csv('2018PSSAEXAMPROCESSED50000TOTALAPPRO.csv')
data.head(5)

sns.set(font_scale=2)

plt.hist(x = data.PercentAdvanced, bins = 10, color = 'green')
plt.title('PercentAdvanced')
plt.xlabel('Score')
plt.ylabel('Number')
plt.show()

plt.hist(x = data.PercentBelowBasic, bins = 10, color = 'red')
plt.title('PercentPercentBelowBasic')
plt.xlabel('Score')
plt.ylabel('Number')
plt.show()

colors = iter(['Uncertain:red purple'])

with sns.axes_style('white'):
   g = sns.factorplot("PercentAdvanced", data=data, aspect=2,
                      kind="count", color='steelblue')
   g.set_xticklabels(step=5)

with sns.axes_style('white'):
   g = sns.factorplot("PercentBelowBasic", data=data, aspect=2,
                      kind="count", color='steelblue')
   g.set_xticklabels(step=5)

sns.pairplot(data = data , hue = 'Population', vars = ['PercentAdvanced'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'DensityAppro',vars = ['PercentAdvanced'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'TotalOffences',vars = ['PercentAdvanced'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'CrimeRateAppro',vars = ['PercentAdvanced'],
palette='Set2',size=10)
```

```
sns.pairplot(data = data , hue = 'GroupAppro',vars = ['PercentAdvanced'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'Rural0Urban1', vars = ['PercentAdvanced'],
palette='Set1',size=10)

sns.pairplot(data = data , hue = 'Population', vars = ['PercentBelowBasic'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'DensityAppro', vars = ['PercentBelowBasic'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'TotalOffences', vars = ['PercentBelowBasic'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'CrimeRateAppro', vars = ['PercentBelowBasic'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'GroupAppro', vars = ['PercentBelowBasic'],
palette='Set2',size=10)

sns.pairplot(data = data , hue = 'Rural0Urban1', vars = ['PercentBelowBasic'],
palette='Set1',size=10)

df = data.copy()
df.head()

x = data.iloc[:,0:10]
y = data.iloc[:,10]
print(x.shape)
print(y.shape)
print(x.columns)

df = data.copy()
df.head()

plt.figure(figsize = (15,12), dpi = 600)
sns.set(font_scale=1.2)
sns.heatmap(data.corr(),annot = True, cmap='Blues')

plt.figure(figsize=(15, 8))
sns.distplot(df.PercentAdvanced)
plt.ylabel('Frequency', fontsize=15)
plt.xlabel('PercenAdvanced', fontsize=15)
plt.title('PercenAdvancedDistribution', fontsize=15)
plt.show()

x = pd.get_dummies(x)
x.head()

from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.24, shuffle=False)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
print(y_test)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
x_train = pd.DataFrame(x_train)
x_train.head()
print(x_train)

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuaracy :", model.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
np.set_printoptions(threshold=50000)
pd.set_option('max_colwidth',1)
print(cm)
print(y_pred)

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
np.set_printoptions(threshold=50000)
pd.set_option('max_colwidth',1)
print(cm)
print(y_pred)

from sklearn.model_selection import cross_val_score
cvs = cross_val_score(estimator = model, X = x_train, y = y_train, cv = 10)
print(cvs)
print(y_pred)

print("Mean Accuracy :", cvs.mean())
print("Variance :", cvs.std())

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
```

```
y_pred = model.predict(x_test)
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
np.set_printoptions(threshold=50000)
pd.set_option('max_colwidth',1)
print(cm)
print(y_pred)

from sklearn.svm import SVC
model = SVC()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
np.set_printoptions(threshold=50000)
pd.set_option('max_colwidth',1)
print(cm)
print(y_pred)

from sklearn.neural_network import MLPClassifier
model = MLPClassifier(hidden_layer_sizes = (100, 100), activation ='relu',
                    solver = 'adam', max_iter = 50)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print("Training Accuracy :", model.score(x_train, y_train))
print("Testing Accuracy :", model.score(x_test, y_test))
cm = confusion_matrix(y_test, y_pred)
np.set_printoptions(threshold=50000, linewidth=1)
pd.set_option('max_colwidth',1)
print(cm)
print(y_pred)
```

Table S2. Coding for feature importance (Python)

```
import pandas as pd
import numpy as np
import subprocess

df = pd.read_csv("2018PSSAEXAMPROCESSED50000APPRO.csv", index_col=0)
ncol = len(df.axes[1])
nrow=len(df.axes[0])
print("%s x %s" % (nrow, ncol)) #check dimension
print (df.dtypes) #check data types

df.groupby('PercentAdvanced').count()

n = ncol
for i in range(0,n):
      if (df.iloc[:,i].dtype==object):
       A= df.iloc[:,i].unique()
       map_to_int = {name: n for n, name in enumerate(A)}
       df.iloc[:,i] = df.iloc[:,i].replace(map_to_int)
print (df.dtypes)


df1=df.iloc[:,0:n]
features = list(df1.columns[:(n-2)])
print("* features:", features, sep="\n")
df1.rename(columns={'y':'Target'}, inplace=True)

y = df1["PercentAdvanced"]
X = df1[features]


%matplotlib

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(20, 20))

df1.hist(bins=20)
plt.show()


corr_df = df1.corr()

%matplotlib inline
import seaborn
import matplotlib.pyplot as plt
print(" CorrelationMatrix")
```

```
mask = np.zeros_like(corr_df)
mask[np.triu_indices_from(mask)] = True
seaborn.heatmap(corr_df, cmap='RdYlGn_r', vmax=1.0, vmin=-1 ,mask = mask,
linewidths=3,fmt='.1f')

plt.yticks(rotation=0,fontsize=10)
plt.xticks(rotation=90,fontsize=10)
plt.show()


print(__doc__)
import numpy as np
from time import time
from operator import itemgetter
from scipy.stats import randint as sp_randint

from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.datasets import load_digits
from sklearn.ensemble import RandomForestClassifier


clf = RandomForestClassifier(n_estimators=20)

def report(grid_scores, n_top=3):
   top_scores = sorted(grid_scores, key=itemgetter(1), reverse=True)[:n_top]
   for i, score in enumerate(top_scores):
      print("Model with rank: {0}".format(i + 1))
      print("Mean validation score: {0:.3f} (std: {1:.3f})".format(
          score.mean_validation_score,
          np.std(score.cv_validation_scores)))
      print("Parameters: {0}".format(score.parameters))
      print("")


param_dist = {"max_depth": [3, None],
          "max_features": sp_randint(1, 11),
          "min_samples_split": sp_randint(1, 11),
          "min_samples_leaf": sp_randint(1, 11),
          "bootstrap": [True, False],
          "criterion": ["gini", "entropy"]}

n_iter_search = 20
random_search = RandomizedSearchCV(clf, param_distributions=param_dist,
                       n_iter=n_iter_search)

start = time()
random_search.fit(X, y)
print("RandomizedSearchCV took %.2f seconds for %d candidates"
```

```
    " parameter settings." % ((time() - start), n_iter_search))
pd.DataFrame(random_search.cv_results_)


from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(criterion='entropy', max_features=3, bootstrap= False,
min_samples_split=3,
                      max_depth=None, min_samples_leaf=10)


forest=clf.fit(X, y)

features_name = np.array(X.columns.values)
features_name

importances = clf.feature_importances_
std = np.std([clf.feature_importances_ for tree in clf.estimators_],
          axis=0)
indices = np.argsort(importances)[::-1]
print("Feature ranking:")

for f in range(X.shape[1]):
   print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

plt.figure()
plt.title("Feature importance",size=20)
plt.bar(range(X.shape[1]), importances[indices],
    yerr=std[indices], color='g' , align="center")
plt.xticks(range(X.shape[1]), features_name[indices],rotation=90,fontsize = 20)
plt.yticks(fontsize = 20)
plt.xlim([-1, X.shape[1]])
plt.show()
```

Table S3. Coding for PA heatmap (R, Population, *10’000)

```
library(tidyverse)
library(readr)
library(maps)
LicenseListWithSecondaries <-
read_csv("C:/Users/DING0/Downloads/20220724PopulationPA.csv")

License_final <- LicenseListWithSecondaries %>%
                  filter(Status == 'Active') %>%
                  mutate(County_join = tolower(str_remove_all(County, " County"))) %>%
                  group_by(County_join) %>% summarise(Frequencies = n())

#m <- map("county","Pennsylvania")

Pennsylvania <- map_data("county","Pennsylvania")

Pennsylvania_final <- inner_join(Pennsylvania, License_final,by=c('subregion' =
'County_join'))

pen_base <- ggplot(data = Pennsylvania_final, mapping = aes(x = long, y = lat, group =
subregion)) +
coord_fixed(1.3) +
geom_polygon(color = "black", fill = "gray")

ditch_the_axes <- theme(
axis.text = element_blank(),
axis.line = element_blank(),
axis.ticks = element_blank(),
panel.border = element_blank(),
panel.grid = element_blank(),
axis.title = element_blank()
)

pen_base +
geom_polygon(aes(fill = Frequencies), color = "white") +
geom_polygon(color = "black", fill = NA) +
theme_bw() +
ditch_the_axes +
scale_fill_gradientn(colours = rev(terrain.colors (14)),
            breaks = c(32, 64, 128))
```

Table S4. Coding for PA heatmap (R, Total Offences, *1000)

```
library(tidyverse)
library(readr)
library(maps)
LicenseListWithSecondaries <-
read_csv("C:/Users/DING0/Downloads/20220724TotalOffencesPA.csv")

License_final <- LicenseListWithSecondaries %>%
                  filter(Status == 'Active') %>%
                  mutate(County_join = tolower(str_remove_all(County, " County"))) %>%
                  group_by(County_join) %>% summarise(Frequencies = n())

#m <- map("county","Pennsylvania")

Pennsylvania <- map_data("county","Pennsylvania")

Pennsylvania_final <- inner_join(Pennsylvania, License_final,by=c('subregion' =
'County_join'))

pen_base <- ggplot(data = Pennsylvania_final, mapping = aes(x = long, y = lat, group =
subregion)) +
coord_fixed(1.3) +
geom_polygon(color = "black", fill = "gray")

ditch_the_axes <- theme(
axis.text = element_blank(),
axis.line = element_blank(),
axis.ticks = element_blank(),
panel.border = element_blank(),
panel.grid = element_blank(),
axis.title = element_blank()
)

pen_base +
geom_polygon(aes(fill = Frequencies), color = "white") +
geom_polygon(color = "black", fill = NA) +
theme_bw() +
ditch_the_axes +
scale_fill_gradientn(colours = rev(terrain.colors (14)),
            breaks = c(128, 256, 512))
```

Table S5. Coding for PA heatmap (R, Real)

```
library(tidyverse)
library(readr)
library(maps)
LicenseListWithSecondaries <-
read_csv("C:/Users/DING0/Downloads/20220724ObservationPAAdvanced.csv")

License_final <- LicenseListWithSecondaries %>%
                filter(Status == 'Active') %>%
                mutate(County_join = tolower(str_remove_all(County, " County"))) %>%
                group_by(County_join) %>% summarise(Frequencies = n())

#m <- map("county","Pennsylvania")

Pennsylvania <- map_data("county","Pennsylvania")

Pennsylvania_final <- inner_join(Pennsylvania, License_final,by=c('subregion' =
'County_join'))

pen_base <- ggplot(data = Pennsylvania_final, mapping = aes(x = long, y = lat, group =
subregion)) +
coord_fixed(1.3) +
geom_polygon(color = "black", fill = "gray")

ditch_the_axes <- theme(
axis.text = element_blank(),
axis.line = element_blank(),
axis.ticks = element_blank(),
panel.border = element_blank(),
panel.grid = element_blank(),
axis.title = element_blank()
)

pen_base +
geom_polygon(aes(fill = Frequencies), color = "white") +
geom_polygon(color = "black", fill = NA) +
theme_bw() +
ditch_the_axes +
scale_fill_gradientn(colours = heat.colors (14),
            breaks = c(8, 12, 16),
            trans = "log2")
```

Table S6. Coding for PA heatmap (R, Prediction)

```
library(tidyverse)
library(readr)
library(maps)
LicenseListWithSecondaries <-
read_csv("C:/Users/DING0/Downloads/20220724PredictionPAAdvanced.csv")

License_final <- LicenseListWithSecondaries %>%
                filter(Status == 'Active') %>%
                mutate(County_join = tolower(str_remove_all(County, " County"))) %>%
                group_by(County_join) %>% summarise(Frequencies = n())

#m <- map("county","Pennsylvania")

Pennsylvania <- map_data("county","Pennsylvania")

Pennsylvania_final <- inner_join(Pennsylvania, License_final,by=c('subregion' =
'County_join'))

pen_base <- ggplot(data = Pennsylvania_final, mapping = aes(x = long, y = lat, group =
subregion)) +
coord_fixed(1.3) +
geom_polygon(color = "black", fill = "gray")

ditch_the_axes <- theme(
axis.text = element_blank(),
axis.line = element_blank(),
axis.ticks = element_blank(),
panel.border = element_blank(),
panel.grid = element_blank(),
axis.title = element_blank()
)

pen_base +
geom_polygon(aes(fill = Frequencies), color = "white") +
geom_polygon(color = "black", fill = NA) +
theme_bw() +
ditch_the_axes +
scale_fill_gradientn(colours = heat.colors (14),
            breaks = c(8, 12, 16),
            trans = "log2")
```

Table S7. Coding for PA heatmap (R, CrimeRate)

```
library(tidyverse)
library(readr)
library(maps)
LicenseListWithSecondaries <-
read_csv("C:/Users/DING0/Downloads/20220724CrimeRate.csv")

License_final <- LicenseListWithSecondaries %>%
                  filter(Status == 'Active') %>%
                  mutate(County_join = tolower(str_remove_all(County, " County"))) %>%
                  group_by(County_join) %>% summarise(Frequencies = n())

#m <- map("county","Pennsylvania")

Pennsylvania <- map_data("county","Pennsylvania")

Pennsylvania_final <- inner_join(Pennsylvania, License_final,by=c('subregion' =
'County_join'))

pen_base <- ggplot(data = Pennsylvania_final, mapping = aes(x = long, y = lat, group =
subregion)) +
coord_fixed(1.3) +
geom_polygon(color = "black", fill = "gray")

ditch_the_axes <- theme(
axis.text = element_blank(),
axis.line = element_blank(),
axis.ticks = element_blank(),
panel.border = element_blank(),
panel.grid = element_blank(),
axis.title = element_blank()
)

pen_base +
geom_polygon(aes(fill = Frequencies), color = "white") +
geom_polygon(color = "black", fill = NA) +
theme_bw() +
ditch_the_axes +
scale_fill_gradientn(colours = rev(terrain.colors (14)),
            breaks = c(9, 15, 21))
```

Table S8 Tuning results by random search

param_bootstrap	param_criterion	param_max_depth	param_max_features	param_min_samples_leaf	param_min_samples_split	mean_test_score	std_test_score	rank_test_score
False	entropy	3	2	3	8	0.114589	0.000828	1
True	entropy	3	6	7	6	0.096399	0.033763	2
True	entropy	3	7	10	3	0.094480	0.034814	3
False	gini	3	5	4	9	0.092531	0.030232	4
False	gini	3	7	5	2	0.086420	0.032037	5
True	entropy	None	7	6	7	0.043402	0.026156	6
False	entropy	None	6	9	8	0.041542	0.025095	7
True	gini	None	8	6	7	0.040331	0.027743	8
True	entropy	None	6	2	10	0.038589	0.025954	9
True	gini	None	2	10	2	0.037734	0.018790	10
True	gini	None	4	8	4	0.035726	0.026884	11
False	entropy	None	4	5	9	0.035253	0.022784	12
False	gini	None	1	10	6	0.034574	0.019973	13
False	gini	None	1	7	9	0.030411	0.019482	14
True	entropy	None	1	2	9	0.029644	0.013548	15
True	entropy	3	2	7	1	NaN	NaN	16
False	gini	3	10	1	6	NaN	NaN	17
False	entropy	None	9	8	10	NaN	NaN	18
False	gini	None	7	9	1	NaN	NaN	19
True	gini	None	9	8	5	NaN	NaN	20

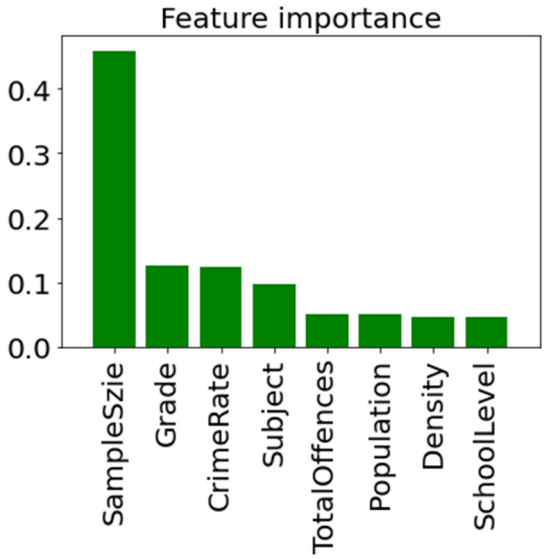

Figure S1. The feature importance of the factors

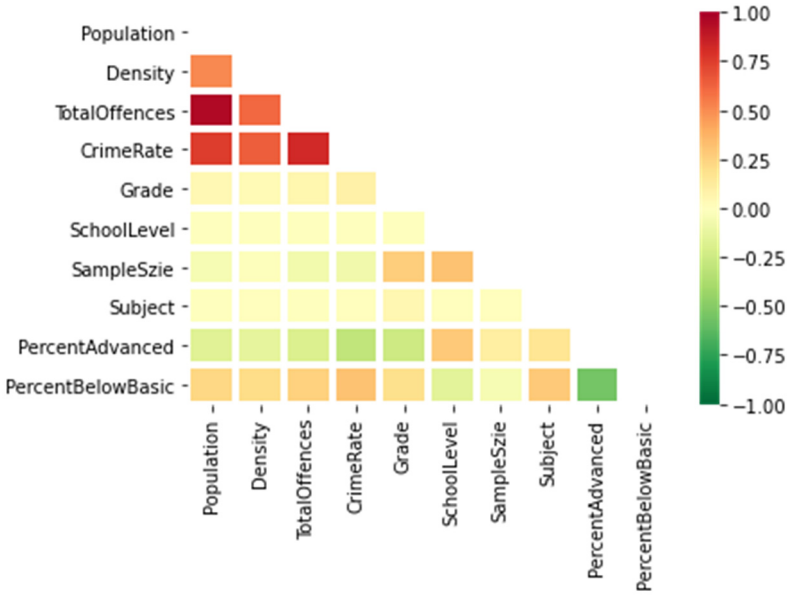

Figure S2. The correlation matrix of the samples