code

March 6, 2022

[2]:
```
# %load ../standard_import.txt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#modeling
import sklearn.linear_model as skl_lm
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.metrics import confusion_matrix, classification_report,␣
 ↪accuracy_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn import preprocessing
from sklearn import neighbors
from sklearn.preprocessing import scale

import statsmodels.api as sm
import statsmodels.formula.api as smf

%matplotlib inline
plt.style.use('seaborn-white')


# Tree plotting
import pydot
from IPython.display import Image
import graphviz
#from sklearn.externals.six import StringIO
from io import StringIO

# Model selection
from sklearn.metrics import mean_squared_error, confusion_matrix,␣
 ↪classification_report, accuracy_score
from sklearn.model_selection import train_test_split, cross_val_score
import folium
```

```
# Trees
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier,␣
 ↪export_graphviz

%matplotlib inline
plt.style.use('seaborn-white')
```

1 Reading and cleaning data

[4]:
```
## THis is the NTC crime stats data from Kaggle
df = pd.read_csv('/Users/fardoussabnur/Desktop/NYC_crime.csv', index_col = 0)
df.head()
```

[4]:
```
   arrest_key arrest_date                                 pd_desc  \
0   192799737  2019-01-26                           SEXUAL ABUSE
1   193260691  2019-02-06  CRIMINAL SALE OF A CONTROLLED SUBSTANCE
2   149117452  2016-01-06                                  RAPE 3
3   190049060  2018-11-15                                  RAPE 1
4    24288194  2006-09-13                     TRESPASS 3, CRIMINAL

                        ofns_desc    law_code law_cat_cd age_group perp_sex  \
0                      SEX CRIMES  PL 1306503          F     45-64        M
1  CONTROLLED SUBSTANCES OFFENSES  PL 2203400          F     25-44        M
2                            RAPE  PL 1302503          F     25-44        M
3                            RAPE  PL 1303501          F     25-44        M
4               CRIMINAL TRESPASS  PL 140100E          M     45-64        M

  perp_race   latitude  longitude arrest_boro  arrest_precinct  \
0     BLACK  40.800694 -73.941109           M               25
1   UNKNOWN  40.757839 -73.991212           M               14
2     BLACK  40.648650 -73.950336           K               67
3     BLACK  40.674583 -73.930222           K               77
4     BLACK  40.671254 -73.926714           K               77

   jurisdiction_code  :@computed_region_f5dn_yrer  \
0                0.0                          7.0
1                0.0                         12.0
2                0.0                         61.0
3                0.0                         16.0
4                2.0                         16.0

   :@computed_region_yeji_bk3q  :@computed_region_92fq_4b7q  \
0                          4.0                         36.0
1                          4.0                         10.0
2                          2.0                         11.0
```

```
3                        2.0                           49.0
4                        2.0                           49.0

   :@computed_region_sbqj_enih
0                         16.0
1                          8.0
2                         40.0
3                         49.0
4                         49.0
```

[7]:

```
## these datas are from Census Bureau, I put them all into one dataframe␣
↪manually

second_data =pd.read_excel('/Users/fardoussabnur/Desktop/datasci724/
↪finalproject/Demographics.xlsx', sheet_name= 'Tract')
second_data.head()
```

[7]:

```
         GEOID                                GEO_LABEL  \
0  36005000100   Census Tract 1, Bronx County, New York
1  36005000200   Census Tract 2, Bronx County, New York
2  36005000400   Census Tract 4, Bronx County, New York
3  36005001600  Census Tract 16, Bronx County, New York
4  36005001900  Census Tract 19, Bronx County, New York

   Foreign-Born (# of total population)  Youth Population (# under 18)  \
0                                1057.0                          171.0
1                                1551.0                          960.0
2                                1051.0                         1127.0
3                                1822.0                         1501.0
4                                 606.0                          757.0

   Completed High School or High School and Some College (# of adults 25+)  \
0                                        2976.996310
1                                        2352.904918
2                                        3285.670285
3                                        3348.367408
4                                        1523.093220

   Poverty (# of individuals in households with incomes below poverty)  \
0                                                  0
1                                               1028
2                                                549
3                                               1264
4                                                843

   Total Population (#)
0                  7080
```

```
1                   4542
2                   5634
3                   5917
4                   2765
```

[8]:
```
second_data.shape
```

[8]:
```
(2166, 7)
```

The following code snippet converts the longitude and latitude to census tracts to match the second dataset

[9]:
```
import geopandas as gpd
import pandas as pd
from shapely.geometry import Point
import os

os.environ['PROJ_LIB'] = r'C:\Users\root\Anaconda3\Library\share\proj'
census_tracts = gpd.read_file('/Users/fardoussabnur/Downloads/tl_2021_36_tract/
↪tl_2021_36_tract.shp')

#points_df = pd.read_csv(r'''C:\Users\root\Desktop\Desktop\airbnb_LAarea.
↪csv''', index_col=0)

geometry = [Point(xy) for xy in zip(df.longitude, df.latitude)]
crs = {'init' :'epsg:4326'}
gdf = gpd.GeoDataFrame(df, crs=crs, geometry=geometry)

merged_file = gpd.sjoin(gdf, census_tracts, how='left', op='within')

merged_df = pd.DataFrame(merged_file)
```

```
/Users/fardoussabnur/opt/anaconda3/lib/python3.8/site-
packages/pyproj/crs/crs.py:131: FutureWarning: '+init=<authority>:<code>' syntax
is deprecated. '<authority>:<code>' is the preferred initialization method. When
making the change, be mindful of axis order changes:
https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-changes-in-
proj-6
  in_crs_string = _prepare_from_proj_string(in_crs_string)
/Users/fardoussabnur/opt/anaconda3/lib/python3.8/site-
packages/IPython/core/interactiveshell.py:3263: FutureWarning: The `op`
parameter is deprecated and will be removed in a future release. Please use the
`predicate` parameter instead.
  if (await self.run_code(code, result,  async_=asy)):
<ipython-input-9-a9d6b48487ac>:15: UserWarning: CRS mismatch between the CRS of
left geometries and the CRS of right geometries.
Use `to_crs()` to reproject one of the input geometries to match the CRS of the
other.
```

```
Left CRS: +init=epsg:4326 +type=crs
Right CRS: EPSG:4269

  merged_file = gpd.sjoin(gdf, census_tracts, how='left', op='within')
```

[10]:
```
geoid = merged_df['GEOID'].nunique()
geoid
```

[10]:
```
2339
```

there are 2,168 census tracts in NYC, but the crime data has a lot of extra tracts

[11]:
```
# Pick all the crimes with arrest dates from 2014 to 2018, bc the ensus

# Convert the date to datetime64
merged_df['arrest_date'] = pd.to_datetime(merged_df['arrest_date'],␣
 ↪format='%Y-%m-%d')

# Filter data between two dates
filtered_df = merged_df.loc[(merged_df['arrest_date'] >= '2014-01-01') &␣
 ↪(merged_df['arrest_date'] <= '2018-12-30')]
```

[12]:
```
# filtering the data to only have selcted columns

filtered_df = filtered_df[['arrest_date', 'ofns_desc', 'age_group', 'perp_sex',␣
 ↪'perp_race','latitude', 'longitude', 'GEOID', 'NAMELSAD' ]]
```

[13]:
```
# Checking for null values

filtered_df.isnull().sum()
```

[13]:
```
arrest_date    0
ofns_desc      0
age_group      0
perp_sex       0
perp_race      0
latitude       0
longitude      0
GEOID          0
NAMELSAD       0
dtype: int64
```

[14]:
```
### Following are all the different types of crimes in the crime dataset

unique_offense = filtered_df['ofns_desc'].unique()
unique_offense
```

[14]:
```
array(['RAPE', 'ASSAULT 3 & RELATED OFFENSES', 'SEX CRIMES', 'THEFT',
       'PROSTITUTION & RELATED OFFENSES', 'DANGEROUS WEAPONS',
       'DANGEROUS DRUGS', 'FRAUDS', 'FORGERY', 'BURGLARY', 'ROBBERY',
       'FORCIBLE TOUCHING', 'TERRORISM', 'FELONY ASSAULT', 'SEX OFFENSES',
       'CRIMINAL TRESPASS', 'F.C.A. P.I.N.O.S.', 'ASSAULT',
       'OFFENSES AGAINST THE PERSON', 'PROSTITUTION OFFENSES',
       'MISCELLANEOUS PENAL LAW', 'GRAND LARCENY', 'PETIT LARCENY',
       'OFFENSES INVOLVING FRAUD', 'OTHER TRAFFIC INFRACTION',
       'THEFT-FRAUD', "BURGLAR'S TOOLS", 'LARCENY', 'ARSON',
       'CONTROLLED SUBSTANCES OFFENSES', 'FRAUDULENT ACCOSTING',
       'OBSTRUCTION OF PUBLIC SERVANTS', 'ESCAPE 3', 'JOSTLING',
       'ADMINISTRATIVE CODE', 'GAMBLING',
       'INTOXICATED & IMPAIRED DRIVING',
       'OFFENSES AGAINST PUBLIC ADMINISTRATION', 'HARRASSMENT 2',
       'MURDER & NON-NEGL. MANSLAUGHTER', 'MOVING INFRACTIONS',
       'HARASSMENT', 'LOITERING', 'OTHER STATE LAWS (NON PENAL LA',
       'VEHICLE AND TRAFFIC LAWS', 'OTHER STATE LAWS',
       'OTHER OFFENSES RELATED TO THEFT',
       'POSSESSION OF STOLEN PROPERTY 5',
       'CRIMINAL MISCHIEF & RELATED OFFENSES',
       'INTOXICATED/IMPAIRED DRIVING',
       'OFF. AGNST PUB ORD SENSBLTY & RGHTS TO PRIV',
       'POSSESSION OF STOLEN PROPERTY', 'ALCOHOLIC BEVERAGE CONTROL LAW',
       'GRAND LARCENY OF MOTOR VEHICLE',
       'OTHER STATE LAWS (NON PENAL LAW)', 'OFFENSES RELATED TO CHILDREN',
       'OFF. AGNST PUB ORD SENSBLTY &', 'DISORDERLY CONDUCT',
       'NEW YORK CITY HEALTH CODE', 'NYS LAWS-UNCLASSIFIED FELONY',
       'UNAUTHORIZED USE OF A VEHICLE 3 (UUV)',
       'CRIMINAL MISCHIEF & RELATED OF', 'KIDNAPPING & RELATED OFFENSES',
       'ADMINISTRATIVE CODES', 'ENDAN WELFARE INCOMP',
       'CHILD ABANDONMENT/NON SUPPORT', 'LOITERING FOR DRUG PURPOSES',
       'OFFENSES AGAINST PUBLIC SAFETY', 'HOMICIDE-NEGLIGENT-VEHICLE',
       'MURDER & NON-NEGL. MANSLAUGHTE', 'OFFENSES AGAINST PUBLIC ADMINI',
       'ANTICIPATORY OFFENSES', 'HOMICIDE-NEGLIGENT,UNCLASSIFIED',
       'ABORTION', 'CHILD ABANDONMENT/NON SUPPORT 1',
       'DISRUPTION OF A RELIGIOUS SERVICE', 'GAMBLING OFFENSES',
       'PARKING OFFENSES', 'MONEY LAUNDERING',
       'LOITERING/GAMBLING (CARDS, DICE, ETC)',
       'UNLAWFUL POSS. WEAP. ON SCHOOL GROUNDS',
       'OFFENSES AGAINST SERVICE ANIMALS', 'HOMICIDE',
       'OFFENSES AGAINST PUBLIC ORDER', 'KIDNAPPING, COERCION',
       'OTHER PUBLIC SAFETY OFFENSES', 'UNDER THE INFLUENCE, DRUGS'],
      dtype=object)
```

[15]:
```
unique_offense.size
```

[15]:
```
87
```

[16]:
```
# COnverting GEOID to be numeric value

filtered_df['GEOID'] = filtered_df['GEOID'].apply(pd.to_numeric,␣
↪errors='coerce')
```

[17]:
```
### Visualizing the Data
```

[18]:
```
import plotly.express as px
fig=px.histogram(filtered_df,
                 x="perp_sex",
                 hover_data=filtered_df.columns,
                 title="Perpetrator Sex"
                )
fig.show()
```

[19]:
```
fig=px.histogram(filtered_df,
                 x="perp_race",
                 color="age_group",
                 hover_data=filtered_df.columns,
                 title="Perpetrator race and age"
                )
fig.show()
```

This graph shows the race and sex of the perpetrator

Now we group the crimes by census tracts. My approach is to count how many crimes occurred in a census tract and group by that to match the format of the second dataset. The second dataset counts the Percentage of people who were born outside of the United States, Percentage of Youth Populationunder the age of 18, Percentage of people adults above the age of 25 who completed High School or High School and Some College, Percentage of households with incomes below poverty)

[20]:
```
crimebygeoid = filtered_df.groupby('GEOID').size().to_frame('size').
↪reset_index()
```

newmerged_data is a new dataset with number of crimes per census tracts

[21]:
```
#m = filtered_df.merge(second_data, on='GEOID', how='outer', suffixes=['',␣
↪'_'], indicator=True)
newmerged_data = pd.merge(crimebygeoid,second_data, how='outer', indicator=True)
```

Number of census tracts in both datasets don't match so I'm dropping the extra tracts from the crime dataset that doesn't appear in the census data. AAnd keeping all the tracts that appear in census data in the crime dataset

[22]:
```
newmerged_data = newmerged_data[newmerged_data["_merge"].str.
↪contains("left_only")==False]
```

[23]:
```python
a = newmerged_data[newmerged_data._merge == 'right_only']
b = newmerged_data[newmerged_data._merge == 'left_only']
```

[24]:
```python
newmerged_data.head(5)
```

[24]:
```
         GEOID   size                                GEO_LABEL  \
0  36005000100  159.0   Census Tract 1, Bronx County, New York
1  36005000200  148.0   Census Tract 2, Bronx County, New York
2  36005000400  254.0   Census Tract 4, Bronx County, New York
3  36005001600  261.0  Census Tract 16, Bronx County, New York
9  36005002300  916.0  Census Tract 23, Bronx County, New York

   Foreign-Born (# of total population)  Youth Population (# under 18)  \
0                                1057.0                          171.0
1                                1551.0                          960.0
2                                1051.0                         1127.0
3                                1822.0                         1501.0
9                                 890.0                         1102.0

   Completed High School or High School and Some College (# of adults 25+)  \
0                                        2976.996310
1                                        2352.904918
2                                        3285.670285
3                                        3348.367408
9                                        2529.497451

   Poverty (# of individuals in households with incomes below poverty)  \
0                                                0.0
1                                             1028.0
2                                              549.0
3                                             1264.0
9                                             2187.0

   Total Population (#) _merge
0                7080.0   both
1                4542.0   both
2                5634.0   both
3                5917.0   both
9                4600.0   both
```

[25]:
```python
### Setting null values to 0 instead of dropping it and changinf column names
```

[26]:
```python
newmerged_data.isnull().sum()
```

[26]:
```
GEOID                                                                         0
size                                                                        139
GEO_LABEL                                                                     0
```

```
Foreign-Born (# of total population)                                   43
Youth Population (# under 18)                                          43
Completed High School or High School and Some College (# of adults 25+)    43
Poverty (# of individuals in households with incomes below poverty)     0
Total Population (#)                                                    0
_merge                                                                  0
dtype: int64
```

[27]:
```
newmerged_data['size'] = newmerged_data['size'].fillna(0)
newmerged_data['Foreign-Born (# of total population)'] =␣
 ↪newmerged_data['Foreign-Born (# of total population)'].fillna(0)
newmerged_data['Youth Population (# under 18)'] = newmerged_data['Youth␣
 ↪Population (# under 18)'].fillna(0)
newmerged_data['Completed High School or High School and Some College (# of␣
 ↪adults 25+)'] = newmerged_data['Completed High School or High School and␣
 ↪Some College (# of adults 25+)'].fillna(0)
newmerged_data['Poverty (# of individuals in households with incomes below␣
 ↪poverty)'] = newmerged_data['Poverty (# of individuals in households with␣
 ↪incomes below poverty)'].fillna(0)
```

[28]:
```
## renaming the columns
newmerged_data = newmerged_data.rename(columns={'size': 'Crimes', 'Foreign-Born␣
 ↪(# of total population)': 'ForeignBorn',
                                               'Youth Population (# under 18)':␣
 ↪'YouthPopulation', 'Completed High School or High School and Some College (#␣
 ↪of adults 25+)': 'Education',
                                               'Poverty (# of individuals in␣
 ↪households with incomes below poverty)':'Poverty' })
```

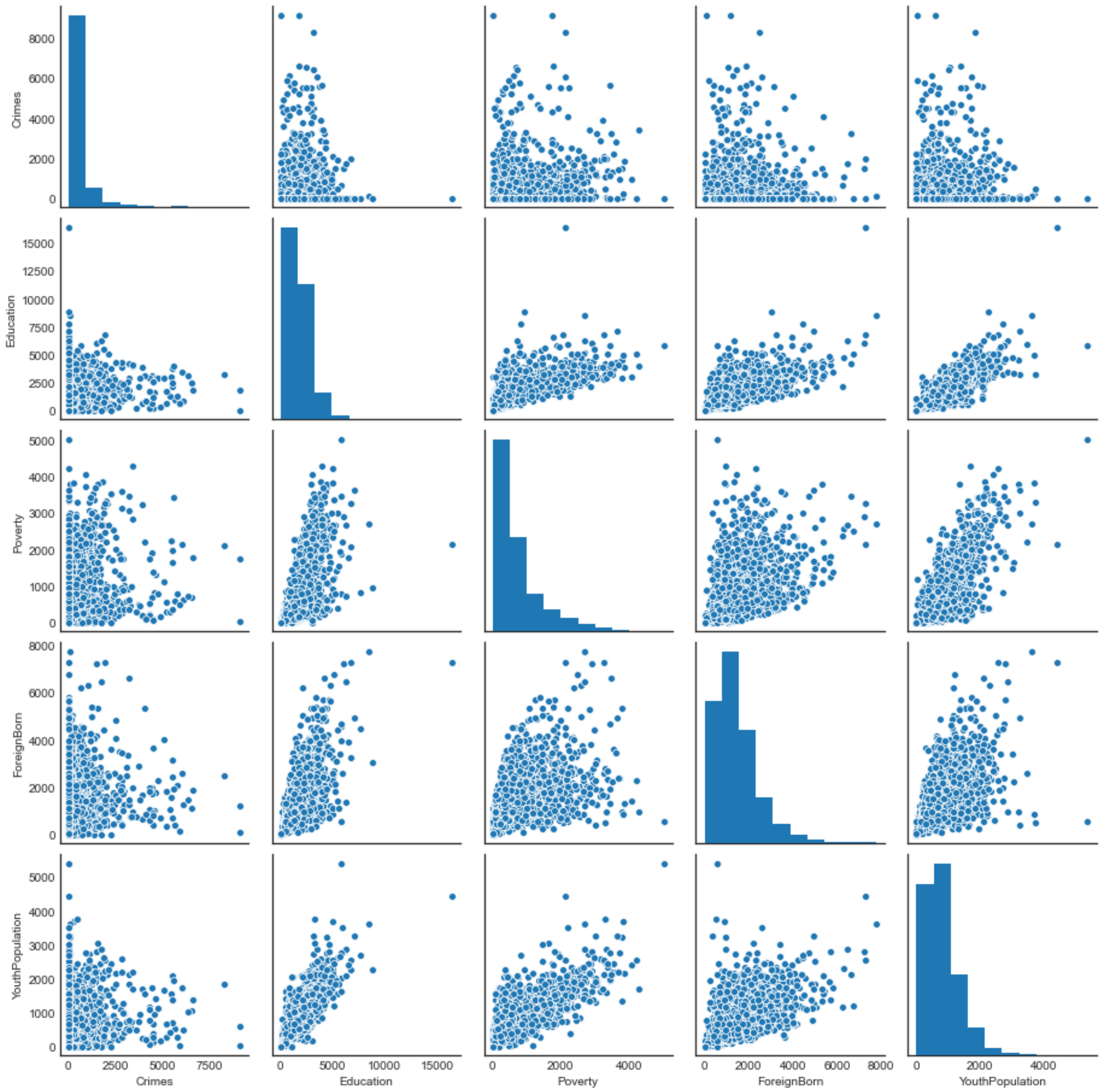
[29]:
```
del newmerged_data['_merge']
```

[30]:
```
newmerged_data.head()
```

[30]:
```
         GEOID  Crimes                              GEO_LABEL  ForeignBorn  \
0  36005000100   159.0   Census Tract 1, Bronx County, New York       1057.0
1  36005000200   148.0   Census Tract 2, Bronx County, New York       1551.0
2  36005000400   254.0   Census Tract 4, Bronx County, New York       1051.0
3  36005001600   261.0  Census Tract 16, Bronx County, New York       1822.0
9  36005002300   916.0  Census Tract 23, Bronx County, New York        890.0

   YouthPopulation    Education  Poverty  Total Population (#)
0            171.0  2976.996310      0.0                7080.0
1            960.0  2352.904918   1028.0                4542.0
2           1127.0  3285.670285    549.0                5634.0
3           1501.0  3348.367408   1264.0                5917.0
9           1102.0  2529.497451   2187.0                4600.0
```

2 Visualizing the new data set

[31]:
```
sns.pairplot(newmerged_data[['Crimes','Education', 'Poverty','ForeignBorn',␣
↪'YouthPopulation']]);
```



[32]:
```
newmerged_data[['Crimes','Education', 'Poverty','ForeignBorn',␣
↪'YouthPopulation']].corr()
```

[32]:

	Crimes	Education	Poverty	ForeignBorn	YouthPopulation
Crimes	1.000000	0.117935	0.286118	0.098041	0.129188
Education	0.117935	1.000000	0.641690	0.670521	0.818343
Poverty	0.286118	0.641690	1.000000	0.484048	0.763665
ForeignBorn	0.098041	0.670521	0.484048	1.000000	0.584357

```
YouthPopulation  0.129188   0.818343  0.763665     0.584357          1.000000
```

2.0.1 Analysis - Multiple Linear Regression

Statsmodel: follows largely the traditional model where we want to know how well a given model fits the data, and what variables "explain" or affect the outcome.

[33]:
```
est = smf.ols('Crimes ~ Education+Poverty+ForeignBorn+YouthPopulation',␣
 ↪newmerged_data)
results = est.fit()
results.summary()
```

[33]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Crimes   R-squared:                       0.101
Model:                            OLS   Adj. R-squared:                  0.099
Method:                 Least Squares   F-statistic:                     60.75
Date:                Sat, 05 Mar 2022   Prob (F-statistic):           1.09e-48
Time:                        23:55:30   Log-Likelihood:                -17615.
No. Observations:                2166   AIC:                         3.524e+04
Df Residuals:                    2161   BIC:                         3.527e+04
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
===
                      coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept         380.7043     34.610     11.000      0.000     312.832
448.576
Education           0.0086      0.030      0.287      0.774      -0.050
0.067
Poverty             0.5467      0.039     14.168      0.000       0.471
0.622
ForeignBorn         0.0044      0.024      0.189      0.850      -0.042
0.051
YouthPopulation    -0.3538      0.066     -5.346      0.000      -0.484
-0.224
==============================================================================
Omnibus:                     2035.553   Durbin-Watson:                   1.781
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            76327.303
Skew:                           4.525   Prob(JB):                         0.00
Kurtosis:                      30.638   Cond. No.                     5.78e+03
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.78e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

Above shows that Poverty and Youth Population are statistically signififacant. Education and Foreign Born is not statistically significant, and if I remove it and run the model again the Adjusted R square increase, and BIC score decreases by a little. Which means when you remove those two variables the model performs better

[34]:
```
est2 = smf.ols('Crimes ~ Poverty+YouthPopulation', newmerged_data).fit()

est2.summary()
```

[34]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Crimes   R-squared:                       0.101
Model:                            OLS   Adj. R-squared:                  0.100
Method:                 Least Squares   F-statistic:                     121.5
Date:                Sat, 05 Mar 2022   Prob (F-statistic):           9.79e-51
Time:                        23:55:30   Log-Likelihood:                -17615.
No. Observations:                2166   AIC:                         3.524e+04
Df Residuals:                    2163   BIC:                         3.525e+04
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
===
                      coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept         386.7309     31.642     12.222      0.000     324.678
448.784
Poverty             0.5478      0.038     14.243      0.000       0.472
0.623
YouthPopulation    -0.3358      0.049     -6.785      0.000      -0.433
-0.239
==============================================================================
Omnibus:                     2034.980   Durbin-Watson:                   1.781
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            76229.027
Skew:                           4.523   Prob(JB):                         0.00
Kurtosis:                      30.619   Cond. No.                     2.46e+03
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.46e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

Scikit-learn Linear Regression: follows the machine learning tradition where the main supported task is chosing the "best" model for prediction.

Running the model on all the variables

[35]:
```
# Regression coefficients (Ordinary Least Squares) - Sciki-Learn
regr = skl_lm.LinearRegression()
X = newmerged_data[[ 'Poverty', 'Education' ,'ForeignBorn', 'YouthPopulation']].
 ↪values # .to_matrix()
y = newmerged_data.Crimes

# Creating train and test data (default choice is proportion of 80:20 for train:
 ↪test)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)


regr.fit(X_train,y_train)
print(regr.intercept_)
print(regr.coef_)
```

```
389.75649485986764
[ 0.52400087 -0.02007256  0.00237252 -0.28841975]
```

[36]:
```
y_pred = regr.predict(X_test)
r_squared = regr.score(X_test, y_test)
```

[37]:
```
r_squared
```

[37]: 0.12577569526792043

[38]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
```

[38]: 643965.223086283

Running the model on the variables that were statistically significant

[39]:
```
# Regression coefficients (Ordinary Least Squares) - Sciki-Learn
regr2 = skl_lm.LinearRegression()
X_sig = newmerged_data[[ 'Poverty', 'YouthPopulation']].values # .to_matrix()
y_sig = newmerged_data.Crimes

# Creating train and test data (default choice is proportion of 80:20 for train:
 ↪test)
X_train_sig, X_test_sig, y_train_sig, y_test_sig = train_test_split(X_sig,␣
 ↪y_sig, random_state=1)


regr2.fit(X_train_sig,y_train_sig)
print(regr2.intercept_)
print(regr2.coef_)
```

```
382.6146433573058
[ 0.52240338 -0.31703836]
```

[40]:
```
y_pred_sig = regr2.predict(X_test_sig)
r_squared = regr2.score(X_test_sig, y_test_sig)
```

[41]:
```
r_squared
```

[41]: 0.12803821873338517

[42]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test_sig, y_pred_sig)
```

[42]: 642298.6182798397

[ ]:

[43]:
```
### The accuracy score increased and Mean squared error decreased when I␣
 ↪removed the variable that were not statistically significant
```

2.1 Ridge regression

Running the model on all the variables

[44]:
```
import sklearn.preprocessing as prepro
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler().fit(X_train)
```

[45]:
```
from sklearn.linear_model import LinearRegression, Ridge, RidgeCV, Lasso,␣
 ↪LassoCV

ridge = Ridge(alpha=len(X)*200)
```

```
ridge.fit(scaler.transform(X_train), y_train)
```

[45]: Ridge(alpha=433200)

[46]:
```
pred = ridge.predict(scaler.transform(X_test))
mean_squared_error(y_test, pred)
```

[46]: 737411.8190757678

[47]:
```
r_squared = ridge.score(X_test, y_test)
r_squared
```

[47]: -7.4472708666619365

[48]:
```
print(pd.Series(ridge.coef_, index = [ 'Poverty', 'Education' ,'ForeignBorn',␣
 ↪'YouthPopulation']))
```

```
Poverty            0.890104
Education          0.350065
ForeignBorn        0.288111
YouthPopulation    0.414111
dtype: float64
```

Running the model on the variables that were statistically significant¶

[49]:
```
scaler2 = StandardScaler().fit(X_train_sig)
```

[50]:
```
ridge2 = Ridge(alpha=len(X_sig)*200)
ridge2.fit(scaler2.transform(X_train_sig), y_train_sig)
```

[50]: Ridge(alpha=433200)

[51]:
```
pred_sig = ridge2.predict(scaler2.transform(X_test_sig))
mean_squared_error(y_test_sig, pred_sig)
```

[51]: 737569.7485957022

[52]:
```
r_squared = ridge2.score(X_test_sig, y_test_sig)
r_squared
```

[52]: -1.6960462611879348

Ridge regression has higjer MSE than linear regression. Here Running the model on the variables that were statistically significant results in higher MSE

2.2 Regression Trees

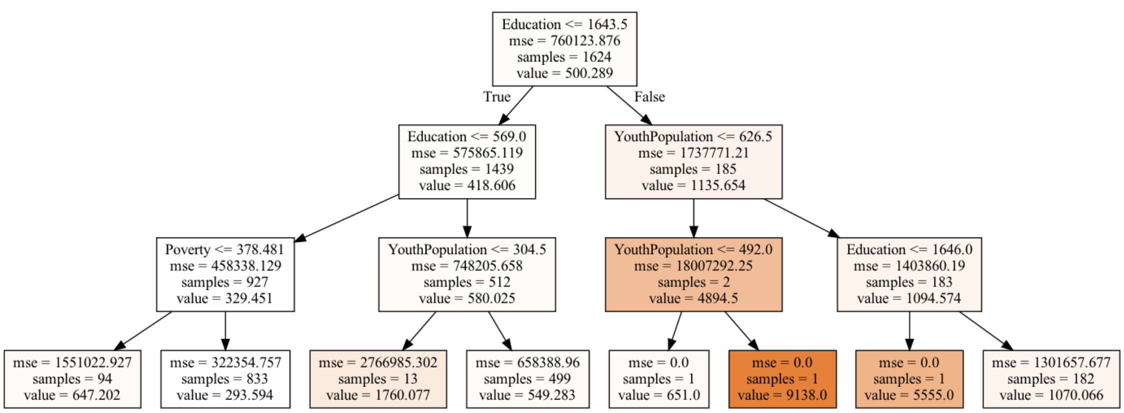
Running the model on all the variables¶

[53]:
```
# Tree with 3 leaf nodes. Note that we fix the random state (seed) for␣
↪reproducibility.

regrTree = DecisionTreeRegressor(max_depth=3, random_state=0)
regrTree.fit(X_train, y_train)
pred = regrTree.predict(X_test)
```

[54]:
```
def print_tree(estimator, features, class_names=None, filled=True):
    tree = estimator
    names = features
    color = filled
    classn = class_names

    dot_data = StringIO()
    export_graphviz(estimator, out_file=dot_data, feature_names=features,␣
↪class_names=classn, filled=filled)
    graph = pydot.graph_from_dot_data(dot_data.getvalue())
    return (graph)
```

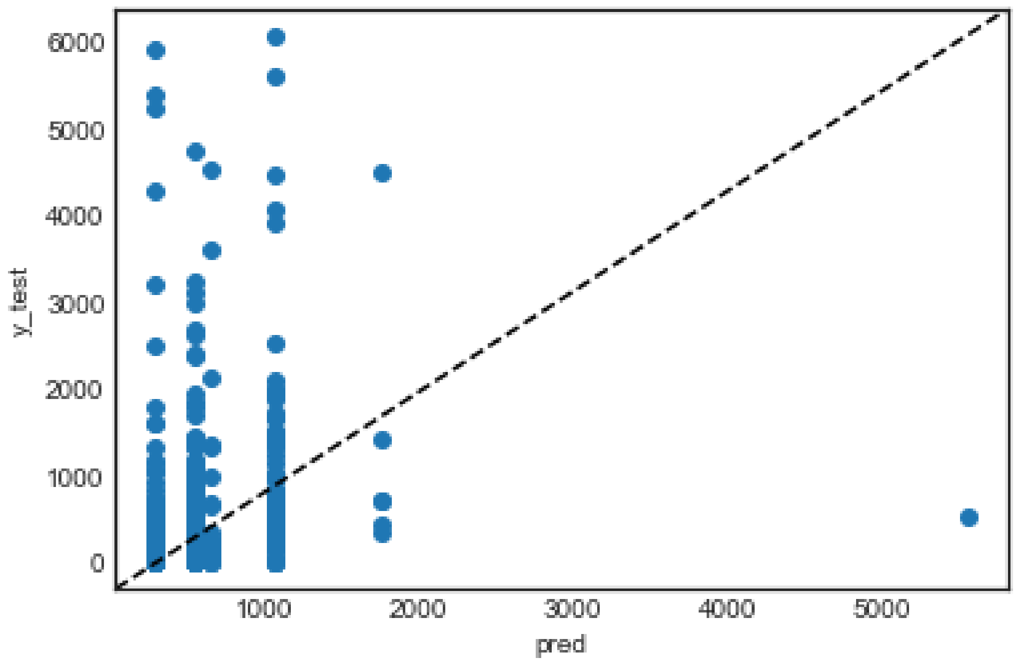
[55]:
```
# Simple plot
import sklearn.tree as tree
# Draw graph b
graph, = print_tree(regrTree, features= ['Education', 'Poverty', 'FOreignBorn',␣
↪'YouthPopulation'])
Image(graph.create_png())
```

[55]:



[56]:
```
plt.scatter(pred, y_test, label='medv')
plt.plot([0, 1], [0, 1], '--k', transform=plt.gca().transAxes)
plt.xlabel('pred')
plt.ylabel('y_test')
```

[56]:
```
Text(0, 0.5, 'y_test')
```



[57]:
```
r_squared = regrTree.score(X_test, y_test)
r_squared
```

[57]: 0.042949179345970534

[58]:
```
# One measure of success
mean_squared_error(y_test, pred)
```

[58]: 704976.3337525371

Running the model on the variables that were statistically significant

[59]:
```
regrTree2 = DecisionTreeRegressor(max_depth=2, random_state=0)
regrTree2.fit(X_train_sig, y_train_sig)
pred2 = regrTree2.predict(X_test_sig)
```

[60]:
```
mean_squared_error(y_test_sig, pred2)
```

[60]: 670354.2362401424

[61]:
```
r_squared = regrTree2.score(X_test_sig, y_test_sig)
r_squared
```

[61]: 0.08995090869569988

2.2.1 Random Forest

Running the model on all the variables

[62]:
```
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier,␣
↪BaggingRegressor, RandomForestRegressor, GradientBoostingRegressor,␣
↪GradientBoostingClassifier
```

[63]:
```
RFmodel = RandomForestClassifier(max_features=3, random_state=0)
RFmodel.fit(X_train, y_train)
```

[63]:
```
RandomForestClassifier(max_features=3, random_state=0)
```

[64]:
```
pred = RFmodel.predict(X_test)
mean_squared_error(y_test, pred)
```

[64]:
```
1179039.9003690036
```

[65]:
```
r_squared = RFmodel.score(X_test, y_test)
r_squared
```

[65]:
```
0.025830258302583026
```

Running the model on the variables that were statistically significant

[66]:
```
RFmodel2 = RandomForestClassifier(max_features=2, random_state=0)
RFmodel2.fit(X_train_sig, y_train_sig)
```

[66]:
```
RandomForestClassifier(max_features=2, random_state=0)
```

[67]:
```
pred2 = RFmodel2.predict(X_test_sig)
mean_squared_error(y_test_sig, pred2)
```

[67]:
```
1154883.721402214
```

[68]:
```
r_squared = RFmodel2.score(X_test_sig, y_test_sig)
r_squared
```

[68]:
```
0.02029520295202952
```

[ ]:
```

```