Práctica 3

Alejandro Borrego Megías, Blanca Cano Camarero

Curso 2020-2021

Índice

Communities and Crime Data Set

Consideraciones previas

- Para ejecutar el problema la estructura de fichero debe de ser la siguiente:

```
|- main.py
|- datos
    |-- communities.data
```

- Todos los tiempos que se muestren estarán en segundos.
- Las prestaciones del equipo en que han sido medidas son:
    - Procesador Intel core i5 de las séptima generación.
    - 8 gigas de ram.

Los datos han sido obtenidos de

https://archive.ics.uci.edu/ml/datasets/communities+and+crime

Descripción del problema

Estamos ante un problema de regresión (aprendizaje supervisado) donde se pretende determinar el número total de crímenes violentos por cada 10^5 habitantes

Esta vez se trata de 1994 instancias que recogen información diversa (raza, renta per cápita, divorciados...) sobre la población de regiones de Estados Unidos,

Los datos que nos da están normalizado, por lo que consideramos la matriz de característica $\mathcal{X} = [0, 1]^{122}$, el vector de etiquetas $\mathcal{Y} = [0, 1]$ y queremos aprender una función $f : \mathcal{X} \to \mathcal{Y}$ que asigne a cada ejemplo la cantidad (normalizada) de crímenes violentos que se producen por cada 100,000 habitantes.

Enfoque elegido para resolver el problema

- Preprocesado de datos (eliminar etributos perdidos, quitar outliers, normalización).
- Intenteto de resulción mediante modelo lineal.
- E intentamos modelos no lineales (SVM, random forest, boosting, MLP).

Interés de la variable

Las atiquetas aportadas son en total 122 predictivas, 5 no predictivas (state, county, community, communityname y fold) y una objetivo (ViolentCrimesPerPop). Para leerlas de forma explícita consultar Attribute Information del apéndice.

Buscando los mejores hiperparámetros.

Codificación de los datos de entrada

Para leer los datos nos enfrentamos a dos problemas:

- Existencia de atributos nominales y no predictivos.
- Hay pérdida de datos.

Atributos no predictivos y nominales

Atendiendo a Attribute Information tenemos que los cinco primeros atributos son no predicctivos, luego los eliminamos directamente en el programa de las prácticas.

El resto de valores son decimales, luego los procesamos sin problema.

Pérdida de datos

Como se indica en la documentación hay datos perdidos, cierta cantidad de datos perdidos en un mismo atributo da lugar a que esta sea de poca utilidad.

Aplicamos el critero dado en el guión, eliminamos los atributos que tengan una pérdida mayor o igual del 20 %, para el resto los completamos con la media de valores válidos de ese atributo más un valor aleatorio dentro del intervalo $[-1{,}5\sigma, 1{,}5\sigma]$ siendo σ la desviación típica de la variable dicha.

Todo esto se implementa en nuestra función `TratamientoDatosPerdidos(x, porcentaje_eliminacion = 20)`.

Separación test y entrenamiento

Utilizamos la separación clásica: 20 % de los datos para test y el resto para entrenamiento , ya que el tamaño de muestra 1994 los consideramos suficiente. Antes de hacer la separación se han desordenado los datos y a partir de ahora solo se trabajará con los datos del conjunto de entrenamiento.

Eliminación de valores atípicos

Vamos consenvar los datos dentro de un intervalo de confianza del 0.997, para eliminar posibles ruidos.

La función utilizada para esto ha sido

```
def EliminaOutliers(y, proporcion_distancia_desviacion_tipica = 3.0):
    '''
    OUTPUT
    (muestra en pantalla alguna información sobre el cálculo de la máscara)
    mascara_y_sin_outliers
    INPUT
    y: etiquetas a las que quitar el outliers

    proporcion_distancia_desviacion_tipica = 3.0

    Información sobre cómo elegir la proporcion_distancia_desviacion_tipica:
    Alguna relaciones:
    distancia | intervalo de confianza:
    1         | 0.683
    1.282     | 0.8
    1.644     | 0.9
    2         | 0.954
    3         | 0.997
    3.090     | 0.998
    4         | 0.9999367
    5         | 0.99999942
    '''
```

Normalizamos los datos

Nos dicen que todos los datos están normalizados al intervalo $[0, 1]$, pero por preprocesados anteriores como haber eliminado outliers es necesario tipificar.

El interés de esto es que algunos de los modelos utilizados son sensibles al escalado de los datos y se recomienda en su documentació normalizarlos primero.

Exiten diferentes métodos de transformación (z-score, min-max, logística...), nosotros hemos optado por el Z-score. ("Sobre normalización En Aprendizaje Automático" n.d.) Que consiste en una transformación de la variable aleatoria X a otra, Z de media cero y varianza uno.

$$Z = \frac{x - \bar{x}}{\sigma}$$

Donde $\bar{x}$ representa la media de X y σ la desviación típica.

Para la implementación utilizamos la función StandardScaler() y los métodos fit_transform(x_train) y scaler.transform(x_test). (**StandardScaler?**)

La necesidad de estos método es normalizar a partir de los datos de entrenamiento, guardar la media y varianza de estos datos y luego aplicar la misma transformación (con los mismo datos de entrenamiento) al test, esto se realiza así ya que si se aplicara la transformación a todos los datos se estaría cometiendo data snopping.

Reducción de dimensión PAC

Las ventajas que nos puede aplicar una reducción PAC es a nivel computacional o para poder visualizar los datos. Puesto que nuestro data set es de tamaño razonable no consideramos conveniente su utilización ya que solo nos restaría en bondad del ajuste.

Tampoco creemos que nos vaya a aportar ninguna ventaja la visualización de los datos por T-SNE, es por ello que prescindimos de ella también.

Procesado aplicado a los datos

Acabamos el prepocesado con los siguientes posibilidades para conjunto de entrenamiento:

- `x_train` : Sin autliers, normalizado.
- `x_train_sin_normalizado`: sin outliers, normalizado.
- `x_train_outliers_normalizado`: con outliers, normalizado.
- `x_train_con_outliers`: con outliers, sin normalizar.

Si nos fijamos en la dimensión de la matriz de características x dependiendo de la transformación utilizada podemos observar lo siguiente:

Tabla 1: Dimensiones obtenidas en X trans preprocesado de los datos

Datos	dimensión
`x_train`	(1556, 100)
`x_train_con_outliers`	(1595, 100)

Como podemos observar, de las 122 características posibles que tenía nuestra matriz x de entrenamiento en un principio, como 5 no son predictivos y la última columna son las etiquetas correspondientes a cada fila, el conjunto de atributos se nos queda en 116.

Después, tras la gestión de atributos perdidos se eliminaron 16 atributos (pues tenían más de un 20 % de valores perdidos), quedándose así un total de 100 características que utilizaremos para entrenar nuestros modelos.

Por otra parte, en el proceso de eliminar outliers (como se puede ver en el número de filas) se eliminan un total de 39 filas.

- SVM con kernel lineal
- Random forest
-

Métrica de error

Para este problema vamos a utilizar el coeficiente de determinación R^2 para analizar la bondad de un ajuste.

El motivo principal es que se encuntra acotado en el intervalo $[0, 1]$ siendo uno el mejor ajuste posible.

Esta métrica se calcula de la siguiente manera:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Donde $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

Cabe destacar que aunque teóricamente el valor está acotado, por le método de cálculo de la propia biblioteca de scikit learn, existen caso en que es arbitrariamente peor. En estos casos nosotros hemos optado por plasmar ese cálculo negativo en la memoria, pero a nivel teórico se considerará que el modelo no explica nada, es decir que $R^2 = 0$.

Modelo Lineal: SVM aplicado a Regresión con Kernel Lineal.

Hemos optado por usar este modelo lineal implementado en la clase `LinearSVR` de `sklearn` usando como función de pérdida la de epsilon_insensitive (que explicaremos más adelante). Una vez entrenado el modelo, como en todos los modelos siguientes, usaremos el coeficiente de determinación R^2 para determinar la bondad del ajuste.

Vamos a optar por no utlizar características polinómicas y tampoco reducir la dimensionalidad del problema, ya que no lo vimos oportuno por el poco número de características y por la información que se podía perder. Finalmente comentar que usaremos los datos

normalizados y sin Outliers, además haremos una transformación polinómica a la matriz de entrenamiento para añadir características cuadráticas.

Estimación de Parámetros

Nota: Este proceso es costoso computacionalmente hablando, y por esto el código se encuentra comentado, para comprobar los valores que se describen aquí abajo descomentar el código:

En este modelo solo vamos a estimar el hiperparámetro **epsilon** y **C**, el primero determina la anchura del pasillo en torno al estimador y C es el término de regularización, ambos se usan en la función de error (epsilon_insensitive). El resto de parámetros los mantendremos por defecto. Así los resultados que obtenemos son los siguientes:

Parámetro C:

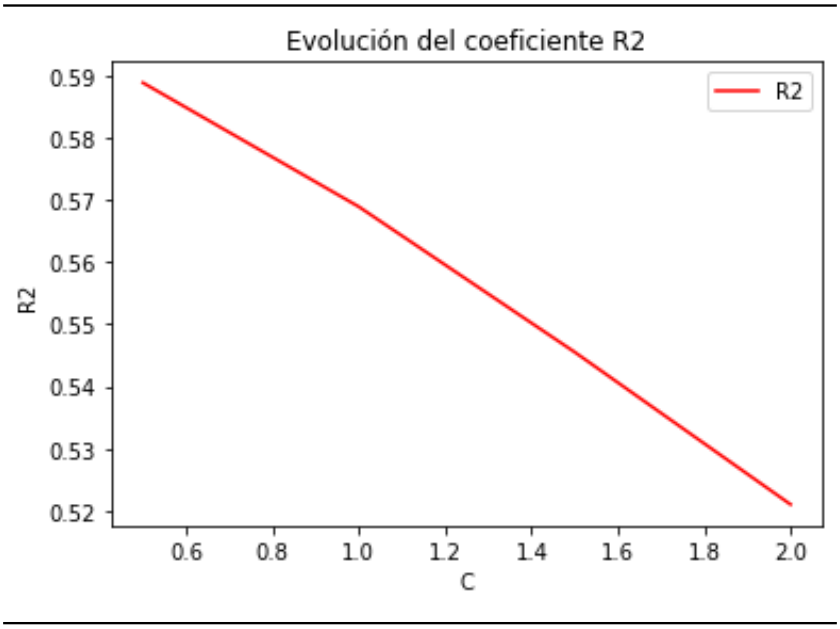
El valor por defecto es 1, así que vamos a tomar valores entorno al 1, como son el 0.5, 1, 1.5 y 2.

Mejores parámetros: {'C': 0.5}

Con una R^2 de: 0.5888467272010589

Tabla 2: Tabla para el parametro C

Parámetros	R^2 medio	Desviación tipica R^2	Ranking	tiempo medio ajuste
C 0.5	0.5888	0.0141	1	34.0404
C 1	0.5689	0.0325	2	35.6532
C 1.5	0.5455	0.0289	3	34.6849
C 2	0.5210	0.0611	4	20.8280

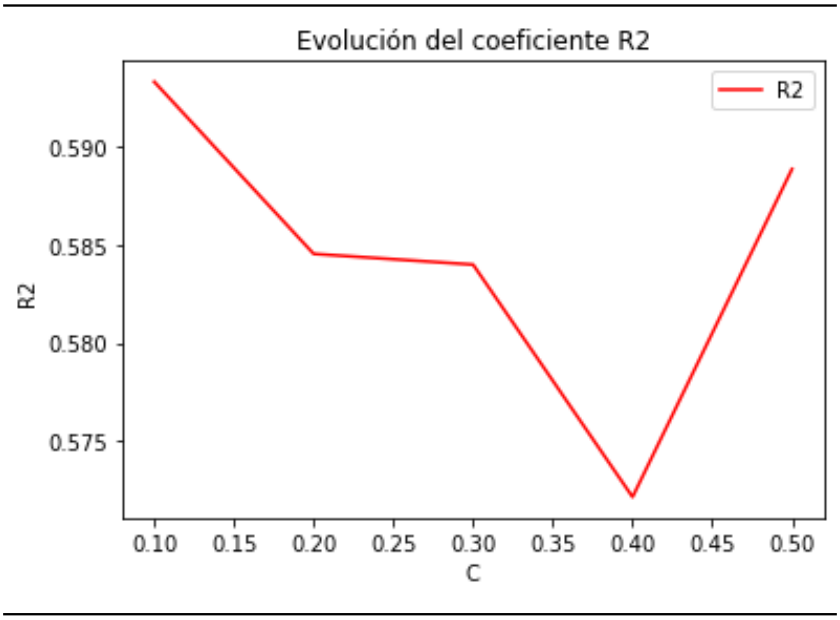


Como podemos observar el máximo parece estar antes del 0.5, por eso vamos a probar entre 0.1 y 0.5 con pasos de 0.1:

Mejores parámetros: {'C': 0.1} Con una R^2 de: 0.5932866031483144

Tabla 4: Tabla para el parametro C

Parámetros	R^2 medio	Desviación tipica R^2	Ranking	tiempo medio ajuste
C 0.1	0.5933	0.0152	1	29.1086
C 0.5	0.5888	0.0141	2	25.9276
C 0.2	0.5845	0.0209	3	30.6160
C 0.3	0.5840	0.0189	4	31.5158
C 0.4	0.5722	0.0367	5	31.6116



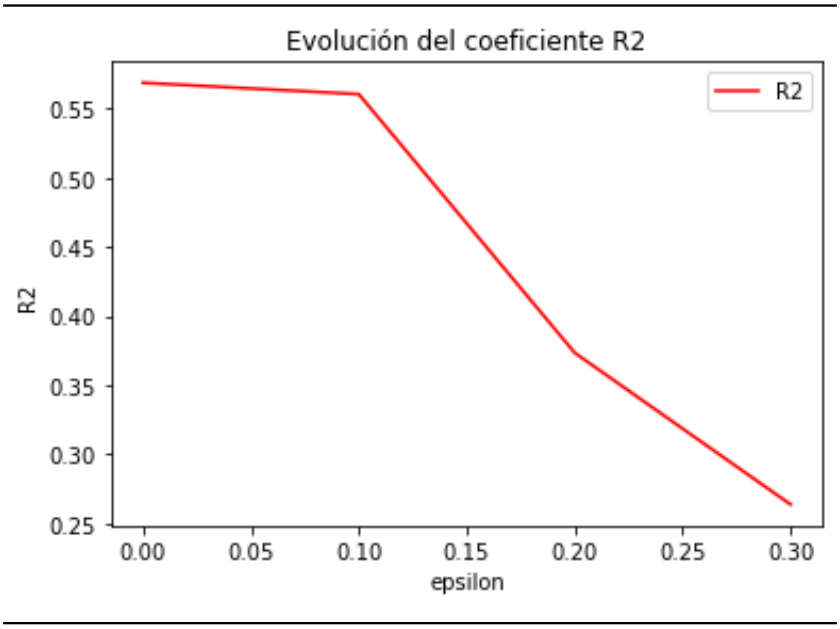
Así, como podemos ver el máximo se alcanza en 0.1, y será ese el valor que tomemos para C

Pasamos ahora a estudiar el parámetro ϵ:

En primero como el valor por defecto es 0, vamos a tomar estos valores: 0.0,0.1,0.2,0.3.

Mejores parámetros: {'epsilon': 0.0} Con una R^2 de: 0.56890579922534

Parámetros	R^2 medio	Desviación tipica R^2	Ranking	tiempo medio ajuste
epsilon 0.0	0.5689	0.0325	1	33.0511
epsilon 0.1	0.5607	0.0303	2	24.7232
epsilon 0.2	0.3734	0.0243	3	8.7525
epsilon 0.3	0.2636	0.0410	4	0.2419



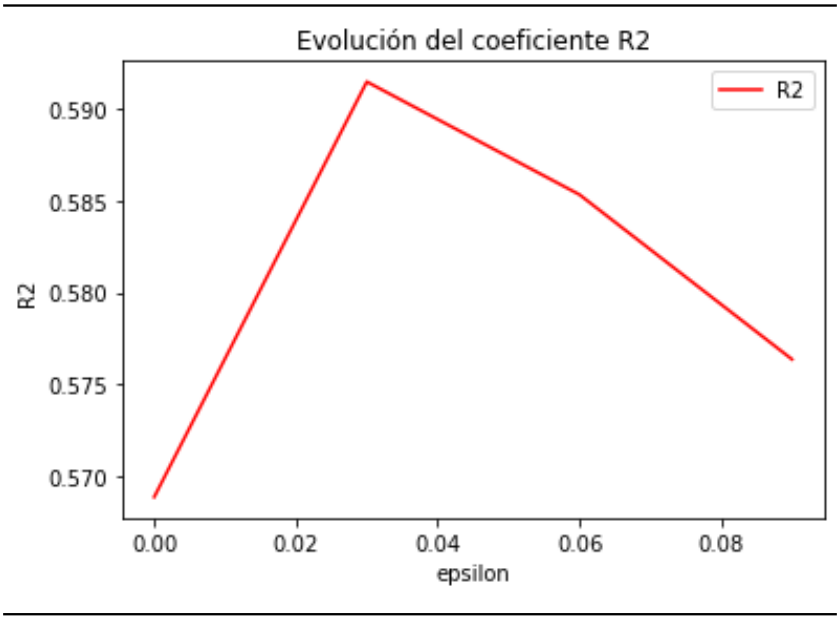
Como vemos la gráfica es decreciente, lo que nos hace pensar que para este modelo lo mejor es que el pasillo sea prácticamente con $\epsilon = 0$. No obstante vamos a probar con algunos valores entre 0 y 0.1, como son 0.03,0.06 y 0.09:

Mejores parámetros: {'epsilon': 0.03}

Con una R^2 de: 0.5914477460146103

Tabla 8: Parámetro ϵ

Parámetros	R^2 medio	Desviación tipica R^2	Ranking	tiempo medio ajuste
epsilon 0.03	0.5914	0.0151	1	35.6540
epsilon 0.06	0.5853	0.0220	2	35.3626
epsilon 0.09	0.5764	0.0213	3	18.8831
epsilon 0.0	0.5689	0.0325	4	34.6573



Como vemos, el máximo se alcanza en $\epsilon = 0{,}03$ luego este será el valor que usemos para nuestro modelo. Tras esto aplicamos nuestro modelo al conjunto de datos de test obteniendo los siguientes resultados:

```
-------------------------------------------------------------
 Evaluando SVM aplicado a Regresión
-------------------------------------------------------------
E_in en entrenamiento:  0.6872076565729107
E_test en validación:  0.6252723796692914
```

Función Pérdida y Regularización

En nuestro caso, el objetivo es minimizar el siguiente problema primal:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1} \max(0, |y_i - (w^T \phi(x_i) + b)| - \varepsilon),$$

como hemos comentado, solo computan aquellos errores fuera del pasillo marcado por ϵ, y C es el término de regularización. En nuestro caso C=0.1 y $\epsilon = 0{,}03$.

Apéndice

Attribute Information

Attribute Information: (122 predictive, 5 non-predictive, 1 goal)
- state: US state (by number) - not counted as predictive above, but if considered, should be consided nominal (nominal)
- county: numeric code for county - not predictive, and many missing values (numeric)
- community: numeric code for community - not predictive and many missing values (numeric)
- communityname: community name - not predictive - for information only (string)
- fold: fold number for non-random 10 fold cross validation, potentially useful for debugging, paired tests - not predictive (numeric)
- population: population for community: (numeric - decimal)
- householdsize: mean people per household (numeric - decimal)
- racepctblack: percentage of population that is african american (numeric - decimal)
- racePctWhite: percentage of population that is caucasian (numeric - decimal)
- racePctAsian: percentage of population that is of asian heritage (numeric - decimal)
- racePctHisp: percentage of population that is of hispanic heritage (numeric - decimal)
- agePct12t21: percentage of population that is 12-21 in age (numeric - decimal)
- agePct12t29: percentage of population that is 12-29 in age (numeric - decimal)
- agePct16t24: percentage of population that is 16-24 in age (numeric - decimal)
- agePct65up: percentage of population that is 65 and over in age (numeric - decimal)
- numbUrban: number of people living in areas classified as urban (numeric - decimal)
- pctUrban: percentage of people living in areas classified as urban (numeric - decimal)
- medIncome: median household income (numeric - decimal)
- pctWWage: percentage of households with wage or salary income in 1989 (numeric - decimal)
- pctWFarmSelf: percentage of households with farm or self employment income in 1989 (numeric - decimal)
- pctWInvInc: percentage of households with investment / rent income in 1989 (numeric - decimal)
- pctWSocSec: percentage of households with social security income in 1989 (numeric - decimal)
- pctWPubAsst: percentage of households with public assistance income in 1989 (numeric - decimal)
- pctWRetire: percentage of households with retirement income in 1989 (numeric - decimal)
- medFamInc: median family income (differs from household income for non-family households) (numeric - decimal)
- perCapInc: per capita income (numeric - decimal)
- whitePerCap: per capita income for caucasians (numeric - decimal)
- blackPerCap: per capita income for african americans (numeric - decimal)

- indianPerCap: per capita income for native americans (numeric - decimal)
- AsianPerCap: per capita income for people with asian heritage (numeric - decimal)
- OtherPerCap: per capita income for people with 'other' heritage (numeric - decimal)
- HispPerCap: per capita income for people with hispanic heritage (numeric - decimal)
- NumUnderPov: number of people under the poverty level (numeric - decimal)
- PctPopUnderPov: percentage of people under the poverty level (numeric - decimal)
- PctLess9thGrade: percentage of people 25 and over with less than a 9th grade education (numeric - decimal)
- PctNotHSGrad: percentage of people 25 and over that are not high school graduates (numeric - decimal)
- PctBSorMore: percentage of people 25 and over with a bachelors degree or higher education (numeric - decimal)
- PctUnemployed: percentage of people 16 and over, in the labor force, and unemployed (numeric - decimal)
- PctEmploy: percentage of people 16 and over who are employed (numeric - decimal)
- PctEmplManu: percentage of people 16 and over who are employed in manufacturing (numeric - decimal)
- PctEmplProfServ: percentage of people 16 and over who are employed in professional services (numeric - decimal)
- PctOccupManu: percentage of people 16 and over who are employed in manufacturing (numeric - decimal)
- PctOccupMgmtProf: percentage of people 16 and over who are employed in management or professional occupations (numeric - decimal)
- MalePctDivorce: percentage of males who are divorced (numeric - decimal)
- MalePctNevMarr: percentage of males who have never married (numeric - decimal)
- FemalePctDiv: percentage of females who are divorced (numeric - decimal)
- TotalPctDiv: percentage of population who are divorced (numeric - decimal)
- PersPerFam: mean number of people per family (numeric - decimal)
- PctFam2Par: percentage of families (with kids) that are headed by two parents (numeric - decimal)
- PctKids2Par: percentage of kids in family housing with two parents (numeric - decimal)
- PctYoungKids2Par: percent of kids 4 and under in two parent households (numeric - decimal)
- PctTeen2Par: percent of kids age 12-17 in two parent households (numeric - decimal)
- PctWorkMomYoungKids: percentage of moms of kids 6 and under in labor force (numeric - decimal)
- PctWorkMom: percentage of moms of kids under 18 in labor force (numeric - decimal)
- NumIlleg: number of kids born to never married (numeric - decimal)
- PctIlleg: percentage of kids born to never married (numeric - decimal)
- NumImmig: total number of people known to be foreign born (numeric - decimal)
- PctImmigRecent: percentage of *immigrants* who immigated within last 3 years (numeric - decimal)
- PctImmigRec5: percentage of *immigrants* who immigated within last 5 years (numeric - decimal)

- PctImmigRec8: percentage of *immigrants* who immigated within last 8 years (numeric - decimal)
- PctImmigRec10: percentage of *immigrants* who immigated within last 10 years (numeric - decimal)
- PctRecentImmig: percent of *population* who have immigrated within the last 3 years (numeric - decimal)
- PctRecImmig5: percent of *population* who have immigrated within the last 5 years (numeric - decimal)
- PctRecImmig8: percent of *population* who have immigrated within the last 8 years (numeric - decimal)
- PctRecImmig10: percent of *population* who have immigrated within the last 10 years (numeric - decimal)
- PctSpeakEnglOnly: percent of people who speak only English (numeric - decimal)
- PctNotSpeakEnglWell: percent of people who do not speak English well (numeric - decimal)
- PctLargHouseFam: percent of family households that are large (6 or more) (numeric - decimal)
- PctLargHouseOccup: percent of all occupied households that are large (6 or more people) (numeric - decimal)
- PersPerOccupHous: mean persons per household (numeric - decimal)
- PersPerOwnOccHous: mean persons per owner occupied household (numeric - decimal)
- PersPerRentOccHous: mean persons per rental household (numeric - decimal)
- PctPersOwnOccup: percent of people in owner occupied households (numeric - decimal)
- PctPersDenseHous: percent of persons in dense housing (more than 1 person per room) (numeric - decimal)
- PctHousLess3BR: percent of housing units with less than 3 bedrooms (numeric - decimal)
- MedNumBR: median number of bedrooms (numeric - decimal)
- HousVacant: number of vacant households (numeric - decimal)
- PctHousOccup: percent of housing occupied (numeric - decimal)
- PctHousOwnOcc: percent of households owner occupied (numeric - decimal)
- PctVacantBoarded: percent of vacant housing that is boarded up (numeric - decimal)
- PctVacMore6Mos: percent of vacant housing that has been vacant more than 6 months (numeric - decimal)
- MedYrHousBuilt: median year housing units built (numeric - decimal)
- PctHousNoPhone: percent of occupied housing units without phone (in 1990, this was rare!) (numeric - decimal)
- PctWOFullPlumb: percent of housing without complete plumbing facilities (numeric - decimal)
- OwnOccLowQuart: owner occupied housing - lower quartile value (numeric - decimal)
- OwnOccMedVal: owner occupied housing - median value (numeric - decimal)
- OwnOccHiQuart: owner occupied housing - upper quartile value (numeric - decimal)
- RentLowQ: rental housing - lower quartile rent (numeric - decimal)
- RentMedian: rental housing - median rent (Census variable H32B from file STF1A)

(numeric - decimal)
- RentHighQ: rental housing - upper quartile rent (numeric - decimal)
- MedRent: median gross rent (Census variable H43A from file STF3A - includes utilities) (numeric - decimal)
- MedRentPctHousInc: median gross rent as a percentage of household income (numeric - decimal)
- MedOwnCostPctInc: median owners cost as a percentage of household income - for owners with a mortgage (numeric - decimal)
- MedOwnCostPctIncNoMtg: median owners cost as a percentage of household income - for owners without a mortgage (numeric - decimal)
- NumInShelters: number of people in homeless shelters (numeric - decimal)
- NumStreet: number of homeless people counted in the street (numeric - decimal)
- PctForeignBorn: percent of people foreign born (numeric - decimal)
- PctBornSameState: percent of people born in the same state as currently living (numeric - decimal)
- PctSameHouse85: percent of people living in the same house as in 1985 (5 years before) (numeric - decimal)
- PctSameCity85: percent of people living in the same city as in 1985 (5 years before) (numeric - decimal)
- PctSameState85: percent of people living in the same state as in 1985 (5 years before) (numeric - decimal)
- LemasSwornFT: number of sworn full time police officers (numeric - decimal)
- LemasSwFTPerPop: sworn full time police officers per 100K population (numeric - decimal)
- LemasSwFTFieldOps: number of sworn full time police officers in field operations (on the street as opposed to administrative etc) (numeric - decimal)
- LemasSwFTFieldPerPop: sworn full time police officers in field operations (on the street as opposed to administrative etc) per 100K population (numeric - decimal)
- LemasTotalReq: total requests for police (numeric - decimal)
- LemasTotReqPerPop: total requests for police per 100K popuation (numeric - decimal)
- PolicReqPerOffic: total requests for police per police officer (numeric - decimal)
- PolicPerPop: police officers per 100K population (numeric - decimal)
- RacialMatchCommPol: a measure of the racial match between the community and the police force. High values indicate proportions in community and police force are similar (numeric - decimal)
- PctPolicWhite: percent of police that are caucasian (numeric - decimal)
- PctPolicBlack: percent of police that are african american (numeric - decimal)
- PctPolicHisp: percent of police that are hispanic (numeric - decimal)
- PctPolicAsian: percent of police that are asian (numeric - decimal)
- PctPolicMinor: percent of police that are minority of any kind (numeric - decimal)
- OfficAssgnDrugUnits: number of officers assigned to special drug units (numeric - decimal)
- NumKindsDrugsSeiz: number of different kinds of drugs seized (numeric - decimal)
- PolicAveOTWorked: police average overtime worked (numeric - decimal)

- LandArea: land area in square miles (numeric - decimal)
- PopDens: population density in persons per square mile (numeric - decimal)
- PctUsePubTrans: percent of people using public transit for commuting (numeric - decimal)
- PolicCars: number of police cars (numeric - decimal)
- PolicOperBudg: police operating budget (numeric - decimal)
- LemasPctPolicOnPatr: percent of sworn full time police officers on patrol (numeric - decimal)
- LemasGangUnitDeploy: gang unit deployed (numeric - decimal - but really ordinal - 0 means NO, 1 means YES, 0.5 means Part Time)
- LemasPctOfficDrugUn: percent of officers assigned to drug units (numeric - decimal)
- PolicBudgPerPop: police operating budget per population (numeric - decimal)
- ViolentCrimesPerPop: total number of violent crimes per 100K popuation (numeric - decimal) GOAL attribute (to be predicted)

"Sobre normalización En Aprendizaje Automático." n.d. Accessed June 1, 2021. https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/normalize-data.