Análisis de Datos 2024

Practico 1: Obtención, exploración y pretratamiento de datos



**Integrantes**: Patrick Hobbins, Nicolas Silva, Nicolas Sere, Felipe Ham, Marcos Montenegro



***INDICE***

[1. Carga de Datos 2](#_Toc1413352759)

[2. Resumen de las Variables 2](#_Toc1642309793)

[3. Análisis Exploratorio de los Datos 2](#_Toc1165055184)

[4. Limpieza de Datos 6](#_Toc1254078371)

[5. Normalización de Datos 6](#_Toc331696553)

[6. Comparación de inconsistencias 6](#_Toc1190678980)

# 1. Carga de Datos

El primer paso fue cargar el dataset “permisos\_construccion.csv” para comenzar a trabajar con él. Este archivo tiene información detallada sobre los permisos de construcción en San Francisco (Estados Unidos), como números de permiso, tipos de permiso, fechas de creación, direcciones, entre otros. A partir de aquí, se comenzó la exploración y pretratamiento de los datos.

# 2. Resumen de las Variables

Cada columna en el dataset representa una variable que contiene información específica sobre los permisos de construcción.

Algunas de las variables importantes son:

* **Permit Number:** Número único de identificación del permiso.
* Street Number Suffix: los datos vacíos pueden ser porque no se conocen o no apliquen para esta dirección. Esto se dedujo porque hay muchos vacíos.
* Completed Date: los faltantes de esta columna deberían corresponder a las construcciones que aún no finalizaron.
* Structural Notification: hay una gran cantidad de datos faltantes, lo que significa que la gran mayoría de los permisos no tendrían que cumplir o tener en cuenta alguna notificación estructural.
* **Permit Type:** Código que indica el tipo de permiso emitido.
* **Permit Creation Date:** Fecha en que se creó el permiso.
* **Street Number, Street Name, Street Suffix:** Información relacionada con la dirección donde se realiza la construcción.
* **Completed Date:** Fecha en la que se completó el proyecto (si es aplicable).
* **Block, Lot:** Información sobre la parcela donde se lleva a cabo la construcción.

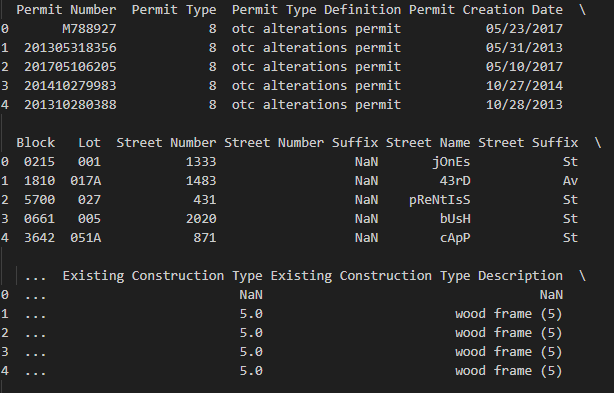
# 3. Análisis Exploratorio de los Datos

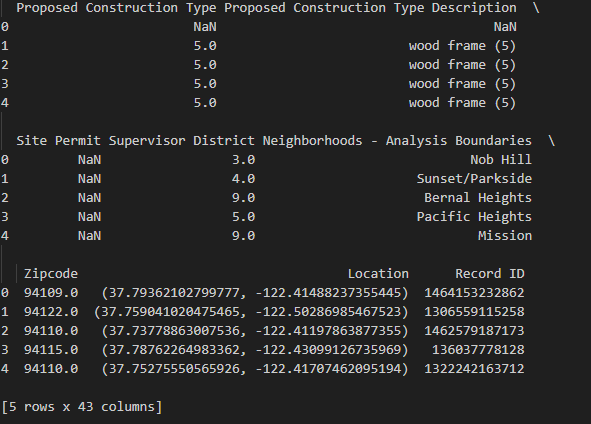
1. **Cantidad de filas y columnas**

El *dataset* está formado por 198910 filas y 43 columnas. Por lo tanto, hay casi 200000 registros divididos en 43 variables.

1. **Observación de las primeras 5 filas**

Luego se obtuvieron las primeras 5 filas para tener una primera vista de la información que tiene cada variable. Además, permite identificar problemas o inconsistencias desde el inicio.





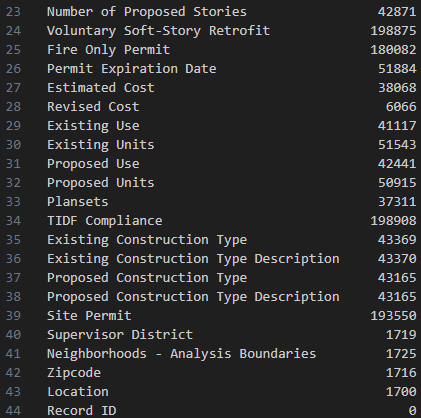
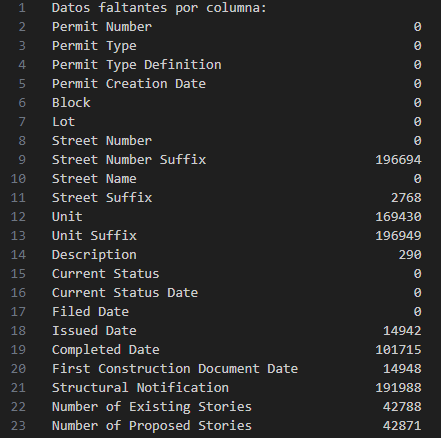
1. **Existencia de datos faltantes y duplicados.**

Durante la exploración, se identificaron datos faltantes en varias columnas:

Las columnas como ‘*Street Number Suffix’, ‘Unit’, y ‘Voluntary Soft-Story Retrofit’* presentan un alto porcentaje de datos faltantes, de los cuales se explicarán posibles motivos en la siguiente parte.

No se encontraron filas duplicadas. Por lo tanto, se puede deducir que cada registro es único.

La cantidad de datos faltantes de cada variable son los siguientes:



1. **Posibles motivos para los datos faltantes.**

Existen varios motivos por los cuales hay datos faltantes, así que a continuación se analizaran los motivos más importantes, en otras palabras, los motivos que abundan más y son más comunes.

Hay varios índices con el mismo motivo, y es que no aplican para este caso, como Street Number Suffix, Street Suffix, Unit Suffix, o Unit.

Mas adelante hay índices como Completed Date, que significa que los faltantes de esta columna deberían corresponder a las construcciones que aún no finalizaron. Luego se encuentran los índices como Structural Notification, lo que quiere decir que hay una gran cantidad de datos faltantes, significando que la gran mayoría de los permisos no tendrían que cumplir o tener en cuenta alguna notificación estructural, o Voluntary Soft-Story Retrofit, que quiere decir que solo unos pocos permisos cuentan con la protección contra terremotos. También hay índices como Fire Only Permit, que quiere decir que son pocos los permisos relacionados a la prevención de incendios. Por último, se encuentra el TIDF Compliance, significando que hay un desconocimiento de este requerimiento legal (por ser nuevo).

1. **Evaluación de valores únicos en variables discretas**

Se evaluaron las variables categóricas. Son aquellas que pueden tomar un conjunto limitado de valores, como el ‘tipo de permiso’ o el ‘sufijo de la calle’. Esto ayuda a comprender la diversidad de valores que hay en cada variable y si algunos valores son más comunes que otros.

1. **Cuantificación de valores únicos y creación de histogramas**

Se cuantificaron los valores únicos de cada variable categórica y se generaron histogramas para poder ver la distribución de estos valores.

Los histogramas son útiles para identificar patrones en los datos, como si algunos tipos de permisos son mucho más comunes que otros.

1. **Evaluación de datos inconsistentes**

Revisamos las columnas de tipo objeto para asegurarnos de que no contengan números, y las columnas de tipo numérico para verificar que no incluyan texto. En un caso específico, encontramos una columna de tipo objeto que contenía un número escrito en palabras. Para corregir esto, convertimos la palabra al número correspondiente y cambiamos el tipo de la columna a numérico.

Otro ejemplo de datos inconsistentes que detectamos fue la mezcla de mayúsculas y minúsculas en palabras dentro de la columna "street name".

Realizaremos este tipo de análisis de forma más detallada para el resto de las columnas en el paso 4.

# 4. Limpieza de Datos

a) . Tomar una decisión respecto a los datos faltantes, duplicados, inconsistencias y cualquier inconveniente que encuentren con los datos.

Con respecto a los datos faltantes, en algunos casos decidimos eliminar las columnas porque cuando una columna tiene demasiados valores ausentes, mantenerla puede aportar poco valor, ya que la información disponible es limitada y podría no ser representativa, lo que podría introducir sesgos si se intenta analizar sobre los datos faltantes. Además, eliminar dichas columnas simplifica el modelo, permitiendo que este se enfoque en las características más relevantes y mejorando así su rendimiento y capacidad de generalización. También se evita la distorsión de resultados que podría ocurrir al rellenar una columna con muchos valores faltantes, especialmente si estos datos no son aleatorios y podrían seguir un patrón que sería difícil de modelar adecuadamente. Por último, al reducir la cantidad de columnas, se mejora la eficiencia computacional, optimizando el tiempo y los recursos necesarios para el análisis, especialmente en conjuntos de datos grandes. Entonces se eliminaron columnas como por ejemplo unit suffix o street number suffix.

En otros casos, lo que hicimos fue rellenar los datos faltantes dependiendo de varios factores, en el caso de los pisos, hay estructuras que no tienen pisos, como los puentes, en ese caso, se los puso un –1.

Con los datos duplicados se decidio ver cada columna por separado, ya que depende de varios factores a tomar en cuenta.

En los datos inconsistentes, como errores tipográficos o formatos mixtos (por ejemplo, direcciones en diferentes formatos), decidimos corregir estos valores para asegurar la uniformidad en el análisis, o sino en casos donde un valor numérico estaba escrito como texto, lo convertimos a su forma numérica y cambiamos el tipo de dato de la columna, ya que esto mejora la precisión y facilita cálculos posteriores, como ya comentado en la parte anterior (3g).

Acá esta explicado específicamente la razón de cada columna:

* Hay que ver qué resolver con estos de Permit Number, en principio, se deja así porque no hay valores NaN, y los repetidos...
* Para la columna ‘Permit Type’ no parecen haber inconsistencias ya que efectivamente el tipo de permiso se representa numericamente y no faltan datos.
* En la columna 'Permit Type Definition', no había valores vacíos, pero sí se encontraron datos inconsistentes, como espacios adicionales o el símbolo '#' en algunos casos. Para los valores que terminaban con '#', se comparó la cantidad de veces que aparecían con y sin este símbolo. Dado que la mayoría de los valores no incluían el '#', se decidió eliminarlo. Además, se eliminaron los espacios en blanco al inicio y al final de los registros, así como los '#' presentes en algunos de ellos.
* Para los valores de la columna 'Permit Creation Date' no se tuvieron que agregar datos ya que no faltaba ninguno. Lo que sí se cambio fue el formato de algunas fechas, ya que algunas estaban con '-' y otras con '/'. Además, a los pocos valores que eran 'NaN' (5), se les colocó la fecha del primer documento de construcción para tener una fecha aproximada.
* Para los valores de la columna ‘Block’, no se cambió nada ya que no faltaban datos y no hay valores 'NaN', pareciendo estar sus valores correctos. Algunos tienen una letra, pero puede llegar a ser parte de su dirección.
* Al igual que la columna anterior 'Block', para 'Lot' no se cambió nada ya que no faltaban datos y no hay valores 'NaN', pareciendo estar sus valores correctos. Algunos tienen una letra, pero puede llegar a ser parte de su dirección.
* Al igual que las columnas anteriores 'Block' y 'Lot', para 'Street Number' no se cambió nada ya que no faltaban datos y no hay valores 'NaN', pareciendo estar sus valores correctos. El tipo de dato es int64 indicando que no hay letras en esta columna de numero de calle.
* Como hay un 98.89 % de datos faltantes del total de filas, se decidió borrar la columna 'Street Number Suffix' ya que no hay manera razonable de ponerle valores y además hay tantos valores faltantes que no aporta valor significativo al analisis.
* Para los valores de la columna 'Street Name' se les corrigió la inconsistencia de variar letras mayúsculas y minúsculas en cada palabra de los nombres de las calles. Se las puso todos con la primera letra mayúscula. Por otro lado, no tenía datos NaN o datos faltantes.
* A los valores NaN de 'Street Name Suffix', que representaban un 1,3% de todos los registros, se le les colocó 'St'. Se dibujó un histograma con la cantidad de veces que aparecía cada uno, y 'St' era significativamente el más predominante. Por lo tanto, al ser pocos valores comparado al total, se asumieron que eran calles (St de Street).
* En la columna ‘Unit’, como puede no aplicar a ciertas construcciones, cambiamos los valores nulos por –1
* En ‘Unit Suffix’ nos encontramos con más del 99% de los datos nulos, por lo que decidimos eliminar esta columna. Decidimos que no había forma de poner valores razonables a tanta cantidad de datos y que no sería significativo para el análisis.
* En la columna ‘Description’ decidimos sustituir los valores nulos por la frase “No description”, que nos pareció lo más razonable
* En ‘Current Status’ no notamos inconsistencias
* Con la columna ‘Number of Existing Stories’, tomamos los valores NaN y los cambiamos por –1, buscando que solo hubiera valores numéricos y que en algunos casos ‘Number of Existing Stories’ no aplicaba a algunos permisos.
* ‘Number of Proposed Stories’ era una columna donde se colocaban los pisos propuestos para el permiso de construcción, en caso de NaN lo remplazamos por los valores de Number Existing Stories, como nos aseguramos de que en la parte anterior los NaN fueran vistos como –1, nos aseguramos de que no quedaran en ‘Number of Proposed Stories’.
* En ‘Voluntary Soft-Story Retrofit’ habian valores “Y” y otros valores que eran NaN, decidimos tomar los “Y” y cambiarlos por True, por otro lado, tomamos los valores NaN como False, la columna paso a ser de tipo bool.
* ‘Fire Only Permit’ fue un caso similar, optamos por cambiar “Y” por True y los NaN por False.
* En el caso de ‘Permit Expiration Date’, tomamos los valores NaN y los cambiamos por la fecha maxima, de manera que en caso de no tener la información poder tener tiempo para formalizar la renovación.
* Para la columna de ‘Estimated Cost’ tomamos la media de los datos y reemplazamos los NaN con ello.
* Referido a ‘Revised Cost’, optamos por tomar los datos de ‘Estimated Cost’ y reemplazar con esos datos los NaN.
* Con ‘Existing Use’ tomamos los valores unicos, buscamos y reemplazamos, esto nos ayudara a posteriori con el ‘Proposed Use’, por otro lado, tomamos los valores NaN y los reemplazamos por “Desconocido”.
* ‘Proposed Use’ son los usos propuestos, en caso de NaN optamos por aplicar los valores de la columna ‘Existing Use’, de esa manera podemos asegurar una mayor correctitud.
* Para la columna de ‘Existing Units’, tomamos los NaN y los cambiamos por la media. Esto con el propósito de no tener una gran desviación en caso de querer generar un modelo de *Machine Learning.*
* ‘Proposed Units’ tuvo el mismo tratamiento que proposed Use, para los NaN tomamos los datos de la columna anterior y remplazamos. Nos aseguramos de que ningún dato sea negativo.

# 5. Normalización de Datos

a) La normalización es el proceso de escalar los valores de las variables numéricas para que se encuentren dentro de un rango específico, como [0, 1] o [-1, 1]. Esto es útil cuando las variables tienen diferentes escalas y quieres garantizar que todas tengan el mismo peso en un análisis.

b) A continuación se analizarán las ventajas de la normalizacion de datos y alguos de sus posibles usos.

En primer lugar, iguala la influencia de las variables en modelos estadísticos y de machine learning, como la regresión lineal, donde variables con rangos más amplios podrían dominar la función de costo. En otras palabras, las variables tienen escalas muy diferentes, las que tienen valores numéricos más altos pueden dominar la influencia en la predicción, independientemente de su importancia real. Por ejemplo, en un modelo que predice el precio de una vivienda, si una variable es el tamaño de la casa en metros cuadrados (que puede variar de 50 a 500 m²) y otra variable es el número de habitaciones (que generalmente varía entre 1 y 10), el tamaño de la casa podría tener un impacto mucho mayor simplemente debido a su escala numérica mayor. La normalización reescala todas las variables a un rango común, asegurando que cada una tenga una influencia proporcional en el modelo, basada en su correlación con la variable objetivo y no en su magnitud original.

Otra ventaja es que en algoritmos de optimizacion, la normalizacion mejora la convergencia. En mas profundidad, los algoritmos de optimización, como el descenso de gradiente, buscan minimizar una función de costo, que mide la diferencia entre las predicciones de un modelo y los valores reales. En términos simples, estos algoritmos ajustan los parámetros del modelo (por ejemplo, los coeficientes en una regresión lineal) para encontrar la combinación que minimiza esta diferencia. Sin la normalizacion, ocurriria que por ejemplo si estás trabajando con un dataset de construcción que incluye varias características de las propiedades, como el tamaño del terreno (en metros cuadrados) y el número de pisos, y supongamos que los valores de tamaño del terreno oscilan entre 100 y 10,000 m², mientras que el número de pisos generalmente varía entre 1 y 5. Sin normalización, la escala de estos dos conjuntos de valores es muy diferente. Como resultado, en la función de costo, el tamaño del terreno tendrá un impacto mucho mayor que el número de pisos, no porque sea más relevante, sino simplemente porque sus valores son numéricamente más grandes.

Cuando el algoritmo de descenso de gradiente intenta minimizar la función de costo, puede verse "distribuido" en una dirección específica debido a esta gran diferencia de escala. Esto causa que el algoritmo haga grandes ajustes en la dirección de las variables de mayor escala (como el tamaño del terreno) y pequeños o ningún ajuste en las de menor escala (como el número de pisos). Esta desproporción en los ajustes puede llevar a que el algoritmo avance lentamente hacia el mínimo de la función de costo o incluso se quede atrapado en un punto sin encontrar la solución óptima.

Con respecto a los posibles usos de la normalizacion, un uso clave de la normalización es en modelos de machine learning, especialmente aquellos que dependen de medidas de distancia. Por ejemplo, en algoritmos como K-Nearest Neighbors, la normalización asegura que todas las variables contribuyan de manera equitativa al cálculo de distancias, evitando que una variable con un rango más amplio domine las decisiones del modelo. En otras palabras,esto es crucial en un contexto de permisos de construcción, donde variables como el salario de los trabajadores y el costo total de un proyecto podrían tener escalas muy diferentes.

Además, la normalización es un paso previo esencial en técnicas de reducción de dimensionalidad como el Análisis de Componentes Principales (PCA). Esta técnica identifica las principales direcciones en las que los datos varían más y reduce el número de variables necesarias para representar los datos sin perder demasiada información. Al normalizar las variables se garantiza que el análisis capture de manera efectiva la estructura de los datos, independientemente de las diferencias de escala entre las variables.

Otro uso importante de la normalización es en la visualización de datos. Al crear gráficos, como diagramas de dispersión, la normalización permite que variables con diferentes unidades de medida sean comparadas de manera clara y efectiva. Esto facilita la identificación de patrones y relaciones en los datos, como la relación entre el salario promedio de los trabajadores y el tamaño de un proyecto de construcción, que de otra manera podrían quedar ocultos debido a las diferencias de escala.

Con respecto al codigo para la normalizacion, se utiliza la clase MinMaxScaler de sklearn para normalizar la variable Salario. Este proceso escala los valores del salario de los trabajadores a un rango entre 0 y 1, lo que facilita la comparación con otras variables que podrían estar en diferentes escalas. Primero se inicializa el escalador, luego el escalador se ajusta a los datos de la variable Salario y se transforma. Por ultimo, los valores normalizados se agregan nuevamente al DataFrame, reemplazando la columna original Salario.

MinMaxScaler() transforma cada valor de una variable continua utilizando la siguiente fórmula:

*Xscaled=X−Xmin/Xmax−Xmin*

Codigo: Normalización de datos:

from sklearn.preprocessing import MinMaxScaler

# Se inicializa el escalador MinMax

scaler = MinMaxScaler()

# Se ajusta y transforman los datos de 'Salario' con el escalador

salario\_normalizado = scaler.fit\_transform(df[['Salario']])

# Se agregan los datos normalizados al dataframe original

df['Salario'] = salario\_normalizado

# 6. Comparación de inconsistencias