UNIVERSIDAD INTERNACIONAL SAN ISIDRO LABRADOR

ESPECIALIZACIÓN DE DATA SCIENCE

PROYECTO NO. 2: MACHINE LEARNING

ULISES JOSÉ BUSTAMANTE MORA

DR. SAMUEL SALDAÑA VALENZUELA

JUNIO, 2024

Contenido

ANTECEDENTES

Pronosticar cambios en el sector de migración es de suma importancia ya que este fenómeno social conlleva consigo múltiples beneficios, según (ONU, 2016):

> "La migración puede propiciar un aumento de la tasa de crecimiento del PIB en los países de destino, el incremento de los salarios de los migrantes, y la expansión de los beneficios indirectos de las remesas para los países de origen."

Explicando la importancia de la migración, si este tema económico lo combinamos con técnicas analísticas, tendríamos un análisis macroeconómico importante para que en ese se basen leyes y normativas fundamentadas en datos y a la vez, para la creación de planes de desarrollo económico y poblacional.

Lo anterior mencionado se podría lograr con la aplicación de técnicas estadísticas junto con software especializado que nos permite analizar una gran cantidad de datos y que este mismo nos arroje el resultado de dicho análisis estadístico para obtener una visión realista sobre la situación y tomar decisiones basadas en datos y no en sesgos.

Dando más fuerza a la idea previamente planteada, según (Leyva, 2023)

> "La estadística constituye una herramienta de gran importancia para la toma de decisiones de trascendencia en nuestras sociedades, y sobre todo en nuestros entornos educativos"

JUSTIFICACIÓN

Partiendo del punto que está más que demostrado las grandes ventajas del análisis de datos para la resolución de problemas, que según (Amazon, s.f.):

> "El análisis de datos ayuda a las empresas a obtener una mayor visibilidad y un conocimiento más profundo de sus procesos y servicios. Les proporciona información detallada sobre la experiencia del cliente y sus problemas. Al cambiar el paradigma más allá de los datos para conectar los conocimientos con la acción, las empresas pueden crear experiencias personalizadas para los clientes y productos digitales relacionados, optimizar las operaciones y aumentar la productividad de los empleados."

Con la aplicación correcta de la tecnología a problemas sociales, podríamos desarrollar ideas de alto impacto a la población que permitan mejorar su calidad de vida. Con esto se obtiene una sociedad sana y segura donde las generaciones, conforme pasan, van agregando más valor agregado gracias a una cultura de conciencia a la política.

OBJETIVO GENERAL

- Predecir conductas migratorias a través del tiempo.

OBJETIVOS ESPECÍFICOS

- Entrenar modelos predictivos con datos limpios para la obtención de futuros resultados.
- Desarrollar conclusiones útiles para la toma de decisiones basadas en estadística.
- Comparar varios algoritmos de machine learning mediante diferentes parámetros para la identificación de la mejor opción.

CRISP-DM

Comprensión del negocio

El objetivo principal de cualquier país es resguardar los derechos fundamentales de todos sus habitantes, esto mediante la creación de políticas que favorezcan dicha misión. En la toma correcta de decisiones para aprobar o rechazar leyes se precisa de datos para corroborar si es factible o no. Esto es de suma importancia ya que puede afectar la calidad de vida de las personas.

El tema de la migración es un tema fundamental para cualquier país, este mismo debe de tener ciertas características que, según el estado sociopolítico de la nación, para que de esta manera sacar lo mejor de las ventajas del fenómeno social de la migración. Para lo toma optima de leyes, se debe tener bajo consideración como esta ha afectado a la nación en épocas anteriores.

En este contexto, se precisa de las técnicas de Data Mining para el análisis de datos como correlaciones, visualización de datos y estadísticos a partir del conjunto de datos previamente recolectados para juzgar en qué periodo se obtuvo mejores resultados y ver sus correlaciones para que de esta manera plantear una solución al problema.

Comprensión de los datos

El conjunto de datos bajo estudio será una recopilación de 1979 hasta el 2015, sobre todos los vuelos, tanto de partida como de llegada del país de Nueza Zelanda, con información extra como el destino y nacionalidad de dichas personas. A la vez, contará con 86525 entradas, sin datos faltantes y donde también destaca la cantidad de personas que se involucran en dichas entradas.

Tomando en cuenta el tipo de dato por variable, se distribuyen de la siguiente manera:

- Measure: Cualitativa nominal
- Country: Cualitativa nominal
- Citizenship: Cualitativa nominal
- Year: Cuantitativa discreta
- Value: Cuantitativa continua

Ilustración 1 Número total de entradas y de datos faltantes y de tipos de datos

Número total de entradas y de datos faltantes y de tipos de datos

```
In [9]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86526 entries, 0 to 86525
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Measure      86526 non-null  object
 1   Country      86526 non-null  object
 2   Citizenship  86526 non-null  object
 3   Year         86526 non-null  int64
 4   Value        86454 non-null  float64
dtypes: float64(1), int64(1), object(3)
memory usage: 3.3+ MB
```

Ilustración 2 Valores únicos entre columnas Measure, Citizenship, Year y Country

Valores únicos en la columna de Measure

```
In [14]: df["Measure"].unique()
```

```
Out[14]: array(['Arrivals', 'Departures', 'Net'], dtype=object)
```

Valores únicos en la columna de Citizenship

```
In [19]: df["Citizenship"].unique()
```

```
Out[19]: array(['New Zealand Citizen', 'Australian Citizen',
       'Total All Citizenships'], dtype=object)
```

Valores únicos en la columna de Year

```
In [20]: df["Year"].unique()
```

```
Out[20]: array([1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989,
       1990, 1991, 1994, 1992, 1993, 1995, 1996, 1997, 1998, 1999, 2000,
       2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011,
       2012, 2013, 2014, 2016, 2015], dtype=int64)
```

Valores únicos en la columna de Country

```
In [18]: df["Country"].unique()
```

```
Out[18]: array(['Oceania', 'Antarctica', 'American Samoa', 'Australia',
       'Cocos Islands', 'Cook Islands', 'Christmas Island', 'Fiji',
       'Micronesia', 'Guam', 'Kiribati', 'Marshall Islands',
       'Northern Mariana Islands', 'New Caledonia', 'Norfolk Island',
       'Nauru', 'Niue', 'New Zealand', 'French Polynesia',
       'Papua New Guinea', 'Pitcairn Island', 'Palau', 'Solomon Islands',
       'French Southern Territories', 'Tokelau', 'Tonga', 'Tuvalu',
       'Vanuatu', 'Wallis and Futuna', 'Samoa', 'Asia', 'Afghanistan',
       'Armenia', 'Azerbaijan', 'Bangladesh', 'Brunei Darussalam',
```

Preparación de los datos

El avistamiento de valores faltantes es crucial para obtener resultados precisos, esto ya que, al haberlos, el modelo estadístico podría interpretarlo de manera incorrecta. Es importante remplazarlos con técnicas como la asignación del promedio o mediana. Lo cual suele funcionar ya que, matemáticamente hablando, no afecta significativamente al modelo.

Ilustración 3 Cambiar los valores faltantes por la mediana

Notar si hay datos faltantes en el conjunto de datos

```
In [26]: df.isnull().any()
```

```
Out[26]: Measure        False
         Country        False
         Citizenship    False
         Year           False
         Value           True
         dtype: bool
```

Se nota que efectivamente hay valores faltantes

```
In [34]: ISNULLfiltro = df["Value"].isnull()
         df[ISNULLfiltro].head()
```

Out[34]:

	Measure	Country	Citizenship	Year	Value
68535	Arrivals	Czechoslovakia	New Zealand Citizen	2009	NaN
68536	Arrivals	Czechoslovakia	Australian Citizen	2009	NaN
68537	Arrivals	Czechoslovakia	Total All Citizenships	2009	NaN
69294	Departures	Czechoslovakia	New Zealand Citizen	2009	NaN
69295	Departures	Czechoslovakia	Australian Citizen	2009	NaN

Se cambia el valor NaN por la mediana de la columna y se comprueba que no hayan más.

```
In [39]: mediana = df["Value"].median()

         df["Value"].fillna(mediana, inplace=True)
```

```
In [38]: df.isnull().any()
```

```
Out[38]: Measure        False
         Country        False
         Citizenship    False
         Year           False
         Value          False
         dtype: bool
```

Otro punto importante es el control de datos atípico, estos datos tienden a cambiar el resultado final negativamente, esto ya que muestra como hecho un valor totalmente alejado de la normalidad, lo cual, a tomarse en cuenta, cambia totalmente la fórmula en el modelo estadístico.

En este caso, se construye un histograma para lograr ver si hay valores normalizados (no los hay ya que a cinco bins se presenta absolutamente una columna, lo cual indica que hay valores atípicos). A la vez, también se dibuja un boxplot con el mismo propósito (sí se presentan valores atípicos.)

Para la construcción de un histograma se precisa de una variable numérica, en este caso sería la columna “Value”. Dado que nos ayuda para la agrupación de todos sus datos y dependiendo de la cantidad de bins, estos se agrupan y visualmente se pueden ver los rangos donde se caen más y menos los datos.

En caso del boxplot, se necesita una variable categórica y una numérica, esto para segmentar y ver en cuales dichos segmentos recaen la mayor cantidad de entradas y de esta manera, también ver la cantidad de datos que se salen de lo común por categoría. Se usarán las columnas “Value” (numérica) y “Measure” (categórica).

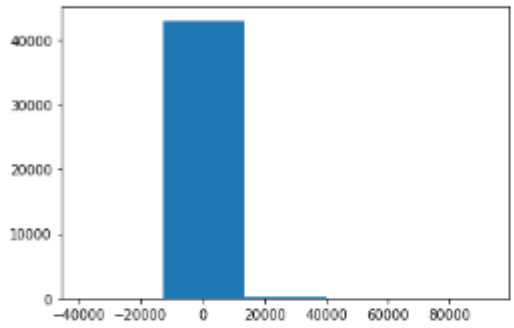
En este caso, no se realizó un remplazo, ya que, dado a las temporadas de viajes a través de año y agentes políticos externos, llega a ser normal que haya una fluctuación de números mayor o menor mediada según la época del año. Se recalca que se realizó un muestreo aleatorio del 50% dado a la longitud del conjunto de datos.

Para esta visualización no se precisó de un tratamiento especial, aunque para mejorar su distribución se pudo realizar una estandarización de los datos, esto cambian su valor bajo una misma desviación y mínimos y máximo, esto ayuda matemáticamente al modelo estadístico ya que los datos estar normalizados.

Ilustración 4 Análisis de datos atípicos

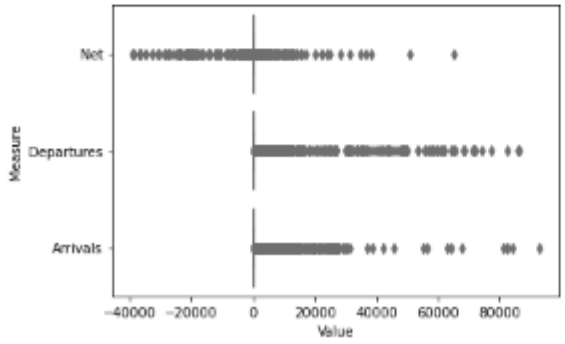
Ver si la columna de Value esta normalizada

```
In [28]: import matplotlib.pyplot as plt

         muestra_aleatoria = df.sample(frac=0.5, random_state=101)

         plt.hist(muestra_aleatoria["Value"], bins=5)
         plt.show()
```



Boxplot para ver si hay datos atipicos

```
In [32]: sns.boxplot(x=muestra_aleatoria["Value"], y = muestra_aleatoria["Measure"], color='skyblue')
Out[32]: <AxesSubplot:xlabel='Value', ylabel='Measure'>
```



Modelado de datos

Saber las medidas de tendencia central es de suma importancia ya que nos brinda de a primera mano información bastante valiosa sobre el dataset, y a partir de estas mismas sacar primeras conclusiones, dentro de ellas se presentan: el promedio y la mediana. A la vez, existen las medidas de dispersión que cumplen también con un papel importante. Estas serían: la varianza y desviación estándar.

Dado a la naturaleza del conjunto de datos, la única variable viable para aplicar las anteriores medidas seria la del total de personas. Se nota otras medidas importantes, como el

conteo, mínimos, máximos y los percentiles. Notamos un promedio bajo a comparación con la desviación estándar, los cual nos indica que los datos no son muy cercanos entre sí.

Para construir dicha tabla, se precisa solamente de una variable numérica.

Ilustración 5 Medidas de tendencia y dispersión

Medidas de tendencia y de disperción

```
In [13]: muestra_aleatoria.describe()["Value"]

Out[13]: count    43263.000000
         mean       241.898320
         std       2979.671533
         min     -39114.000000
         25%          0.000000
         50%          0.000000
         75%          6.000000
         max      92660.000000
         Name: Value, dtype: float64
```

Para calcular la moda en variables, se puede aplicar tanto en variables categóricas como numéricas.

Obteniendo la moda de cada variable

```
In [17]: muestra_aleatoria.mode()
```

Out[17]:

	Measure	Country	Citizenship	Year	Value
0	Net	Ukraine	New Zealand Citizen	2010	0.0

Ilustración 6 Moda de cada variable

Para la mediana, coeficiente de variación, curtosis y tablas cruzadas, solo se aplica a variables numéricas.

Ilustración 7 La mediana de variables numéricas

Cálculo de la mediana

```
In [18]: muestra_aleatoria.median()

Out[18]: Year     1997.0
         Value       0.0
         dtype: float64
```

Ilustración 8 Coeficiente de variación

Cálculo del coeficiente de variación

```
In [21]: muestra_aleatoria["Value"].std() / muestra_aleatoria["Value"].mean()

Out[21]: 12.317867844954
```

Ilustración 9 Curtosis

Curtosis del conjunto de datos

```
In [22]: muestra_aleatoria.kurtosis()

Out[22]: Year      -1.200321
         Value    322.655767
         dtype: float64
```

Ilustración 10 Tabla de agrupación

Agrupacion de la suma de personas, segmentada por la ciudadania y tipo de vuelo

```
In [24]: muestra_aleatoria.pivot_table(values = "Value", index= "Citizenship", columns = "Measure", aggfunc = sum)
```

Out[24]:

Measure	Arrivals	Departures	Net
Citizenship			
Australian Citizen	258471.0	161052.0	63468.0
New Zealand Citizen	1296458.0	2674016.0	-1115723.0
Total All Citizenships	3356208.0	3156397.0	614900.0

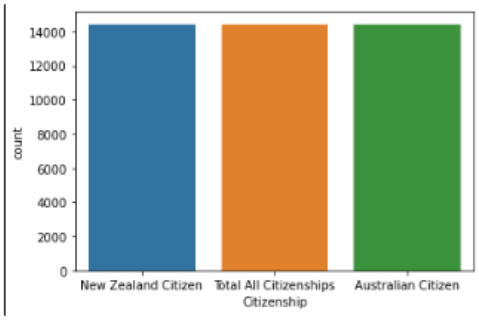
Saber el conteo de las entradas totales por segmentación es una práctica muy importante, esto para notar a primera vista si hay un patrón en su comportamiento, dando como resultado un factor importante a tomar en cuenta. En este caso, cuando segmentamos por nacionalidad y modo de vuelo, se presenta bastante distribuida entre todas las opciones.

Se precisa de una variable categórica o numérica que funcione como categórica para construir este tipo de gráfico, esto ya que hace un conteo por categoría del total de entradas.

Ilustración 11 Conteo por nacionalidad

Conteo de entradas por nacionalidad

In [16]:
```
sns.countplot(x='Citizenship',data=muestra_aleatoria)
plt.show()
```



In [28]:
```
CitiGroupBy = muestra_aleatoria.groupby("Citizenship")["Value"].count()
CitiGroupBy = pd.DataFrame(CitiGroupBy).reset_index()
CitiGroupBy.head()
```

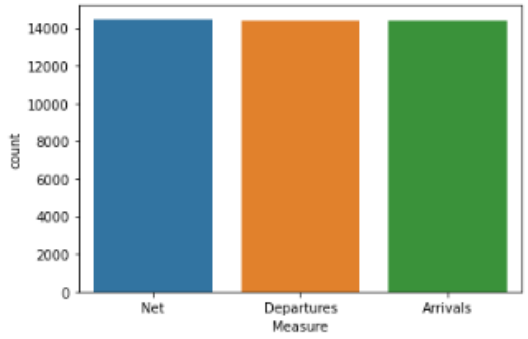
Out[28]:

	Citizenship	Value
0	Australian Citizen	14411
1	New Zealand Citizen	14429
2	Total All Citizenships	14423

Ilustración 12 Conteo por modo de vuelo

Conteo de entradas por modo de vuelo

```
In [29]: sns.countplot(x='Measure',data=muestra_aleatoria)
         plt.show()
```




```
In [27]: MeasureGroupBy = muestra_aleatoria.groupby("Measure")["Value"].count()
         MeasureGroupBy = pd.DataFrame(MeasureGroupBy).reset_index()
         MeasureGroupBy.head()
```

Out[27]:

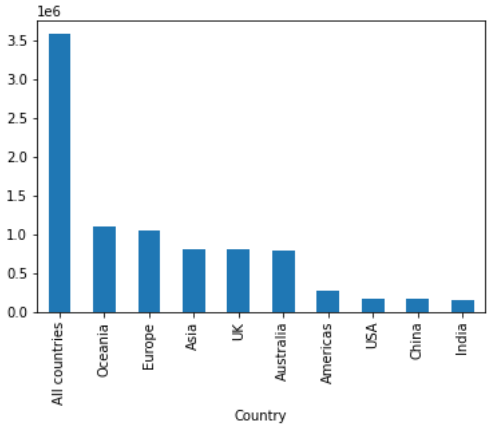
	Measure	Value
0	Arrivals	14373
1	Departures	14411
2	Net	14479

Otro gráfico de barras a considerar es uno en que nos muestre el país o continente donde más movimientos tienen con Nueva Zelanda, esta información nos ayudaría para tener un panorama más claro acerca de la situación actual migratoria, también podría explicar fenómenos internacionales.

Ilustración 13 Países más comunes

Paises mas comunes

```
In [36]: top_10_paises = muestra_aleatoria.groupby("Country")["Value"].sum().sort_values(ascending=False).head(10)
         top_10_paises.plot(kind="bar")
Out[36]: <AxesSubplot:xlabel='Country'>
```



Un tipo de grafico muy importante para agregar, sería el Scatterplot, que nos dice si hay algún tipo de correlación entre dos variables numéricas, saber este dato sería de gran importancia ya que podría analizar a partir de aquí diferentes tipos de mediciones o formulación de hipótesis.

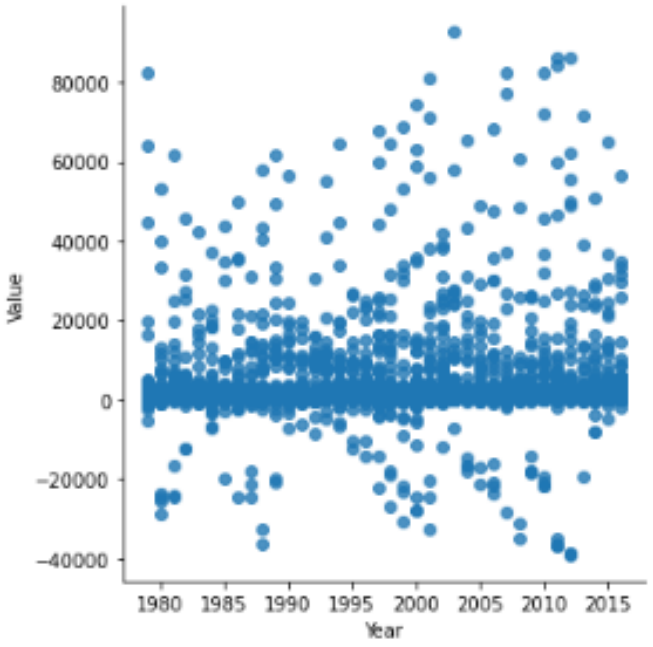
Para nuestro caso, creamos un gráfico con el eje en X el año y en el eje Y el valor. Podríamos notar principalmente una leve correlación ya que la mayoría de los puntos se presentan juntos, es cierto que hay otros que se salen de la regla, pero cabe a rescatar que son una minoría.

Para la construcción del mismo se precisa de dos variables numéricas, esto ya que, al ponerlas en un plano cartesiano, el grafico dibuja un punto donde las dos variables se intersecan, así con todas las entradas. Permitiendo visualmente ver el comportamiento de todos los puntos de conjunto de datos.

Ilustración 14 Scatterplot

Scatterplot para notar algun tipo de correlación

```
In [17]: sns.lmplot(x='Year',y='Value',data=muestra_aleatoria)
Out[17]: <seaborn.axisgrid.FacetGrid at 0x17917d30130>
```

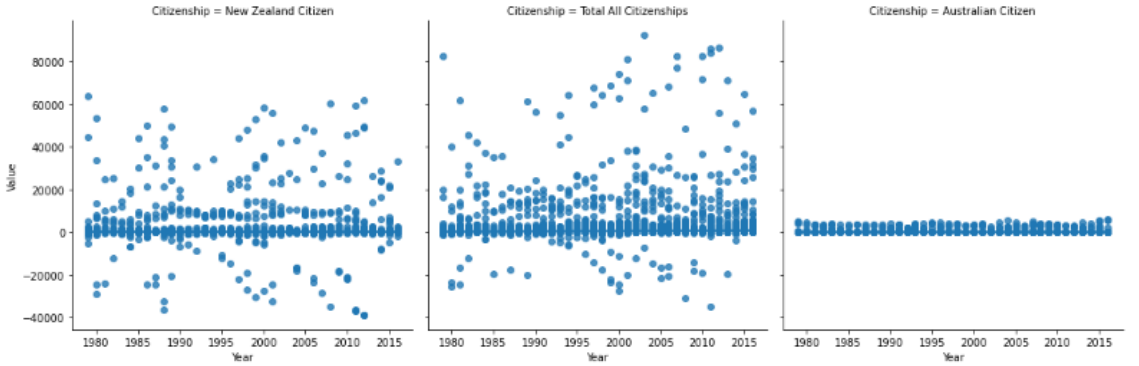


También diseñamos un Scatterplot, pero con la segmentación por ciudadanía para granular más el análisis y notar si alguno de ellos si presentan una mejor visibilidad con respectos a las dos variables a comparar. Notamos que las ciudadanías de Australia y Nueva Zelanda si presentan una mejor correlación.

Para la construcción de este gráfico, se agrega una variable categórica que funciona como filtro para el cálculo de cada uno.

Scatterplot segmentado por ciudadania

```
In [22]: sns.lmplot(x='Year',y='Value',data=muestra_aleatoria,col = "Citizenship")
Out[22]: <seaborn.axisgrid.FacetGrid at 0x17917bc1e80>
```



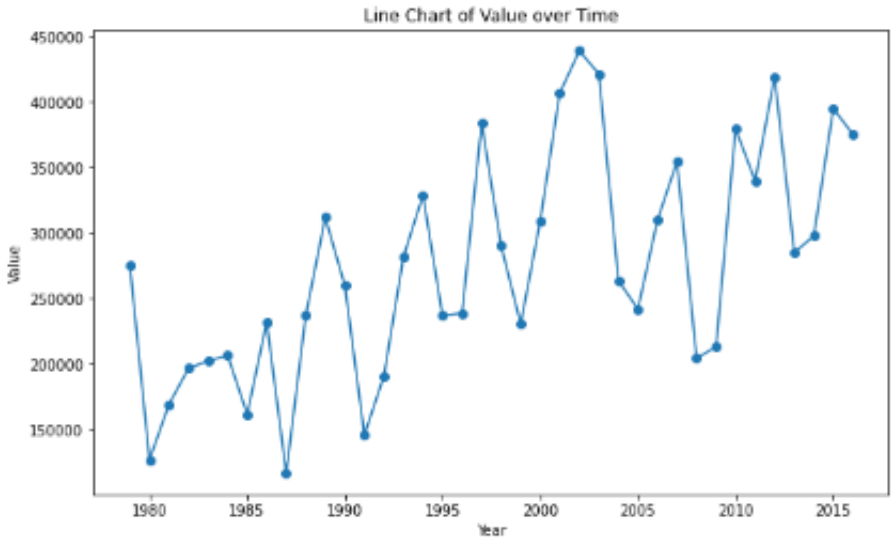
Ya que trabajamos con tiempo, como puede ser la columna de Año, declaramos que sería de gran importancia crear un gráfico de línea para notar visualmente algún tipo de tendencia. Mediante este análisis, notamos una tendencia al alza conforme fueron pasando los años. Este descubrimiento es de alta relevancia ya que nos indicia que en futuros años también podría crecer aún más.

Para construir un lineplot se precisa de una variable categórica o numérica que funcione como categoría, el cual es nuestro caso, pero que hago referencia a una fecha como puede ser días, meses, trimestres, semestres o año. Y una variable numérica que nos ayude a resumir los datos.

Ilustración 15 Lineplot

Lineplot

```
In [19]: plt.figure(figsize=(10, 6))
         plt.plot(YearGroupBy["Year"], YearGroupBy["Value"], marker='o', linestyle='-')
         plt.xlabel('Year')
         plt.ylabel('Value')
         plt.title('Line Chart of Value over Time')

Out[19]: Text(0.5, 1.0, 'Line Chart of Value over Time')
```



Como último paso, se precisaría de un arreglo en el dataset original para que sea totalmente funcional a la hora de aplicar un futuro modelo de Machine Learning, para eso, aplicaremos técnicas de transformación de datos. Entre ellas creación de dummies variables, selección de variables al utilizar y separación del dataset en conjuntos de datos más pequeños.

La creación de dummies se recomienda cuando quisiéramos ingresar una variable categórica como factor a un modelo de ML, el problema sería que dicho modelo es incapaz de interpretar letras, por ende, se debe cambiar dichas características entre números, de esta manera, el modelo podrá leer con satisfacción los datos.

Ilustración 16 Dummies variables

Preparación del dataset para un futuro algoritmo de Machine Learning

Creacion de dummies para las variables categoricas

In [36]:
```
dummies = pd.get_dummies(df['Measure'], drop_first=True)

df_ml = pd.concat([df, dummies], axis=1)
df_ml.head()
```

Out[36]:

	Measure	Country	Citizenship	Year	Value	Departures	Net
0	Arrivals	Oceania	New Zealand Citizen	1979	11817.0	0	0
1	Arrivals	Oceania	Australian Citizen	1979	4436.0	0	0
2	Arrivals	Oceania	Total All Citizenships	1979	19965.0	0	0
3	Arrivals	Antarctica	New Zealand Citizen	1979	10.0	0	0
4	Arrivals	Antarctica	Australian Citizen	1979	0.0	0	0

El siguiente punto es la selección de variables, se recomienda usar las columnas más importantes a la hora de entrenar un modelo de Machine Learning, esto ya que daría mejor resultados, el problema de agregar la máxima cantidad de variables es que podría entorpecer o confundir al modelo. En esta ocasión solo utilizaremos el año y el tipo de vuelo.

Por último, el dataset es dividido entre variables independientes (X) y variables dependientes (y), que esta ultima el dato que nos gustaría predecir en base a las variables independientes. A la vez, estos mismos dos conjuntos de datos se vuelven a dividir, dando un total de cuatro datasets. Estos cuatro serán utilizados a lo largo de modelos de Machine Learning.

Ilustración 17 Selección de variables y separación

Preparación del dataset para entrenar y evaluar el modelo

In [35]:
```
from sklearn.model_selection import train_test_split

X = df_ml[["Year", "Departures", "Net" ]]
y = df_ml["Value"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state = 101)
```

Datasets listos

In [37]:
```
X_train.head()
```

Out[37]:

	Year	Departures	Net
49743	2000	0	1
19360	1987	1	0
40772	1996	0	1
85812	2016	0	1
50068	2000	0	1

In [38]:
```
y_train.head()
```

Out[38]:
```
49743   -2.0
19360    0.0
40772   -2.0
85812   -1.0
50068    0.0
Name: Value, dtype: float64
```

MACHINE LEARNING

Regresión Lineal

Uno de los modelos de machine learning más famosos y útiles es la regresión lineal o en inglés Linear Regression, gracias a su fácil codificación y alto rendimiento, se vuelve la herramienta por excelencia para predecir valores numéricos continuos. Para su uso necesitaríamos una o varias variables independientes (X) y una variable dependiente a predecir (y).

Primer Modelo

Analizando mi conjunto de datos, observados que las variables “Year” y “Value” son numéricas, así que el primer intento será con estas dos variables solas, para encontrar si tienen alguna correlación. Además, se agregó tres filtros, uno por “Measure”, “Value” y otro por “Year” para limitar el número de años que le gustaría agregar.

Ilustración 18 Código de Linear Regression

Creación de un Linear Regression por año para predecir la cantidad de personas con un filtro de metodo de viaje. Con la metrica del RMSE y el Variance Score ¶

```
measure_eleccion = "Arrivals" # Arrivals Departures Net
en_los_ultimos = 15 # Si no quieres filtrar por años, poner 38
thresholder_value = 2000


df_mod1 = df.where(df["Measure"] == measure_eleccion).dropna()
df_mod2 = df_mod1.where(df_mod1["Year"] >= max(df_mod1["Year"]-en_los_ultimos)).dropna()
df_mod = df_mod2.where(df_mod2["Value"] >= thresholder_value).dropna()

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import mean_squared_error

X_lm = df_mod[["Year"]]
y_lm = df_mod["Value"]
X_train_lm, X_test_lm, y_train_lm, y_test_lm = train_test_split(X_lm, y_lm, test_size=0.30, random_state = 101)

lm = LinearRegression().fit(X_train_lm,y_train_lm)
lm_pred = lm.predict(X_test_lm)

print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test_lm, lm_pred)))
print("Variance Score: ", round(100*(metrics.explained_variance_score(y_test_lm, lm_pred)),2),"%")

plt.scatter(y_test_lm, lm_pred)
plt.show()
```

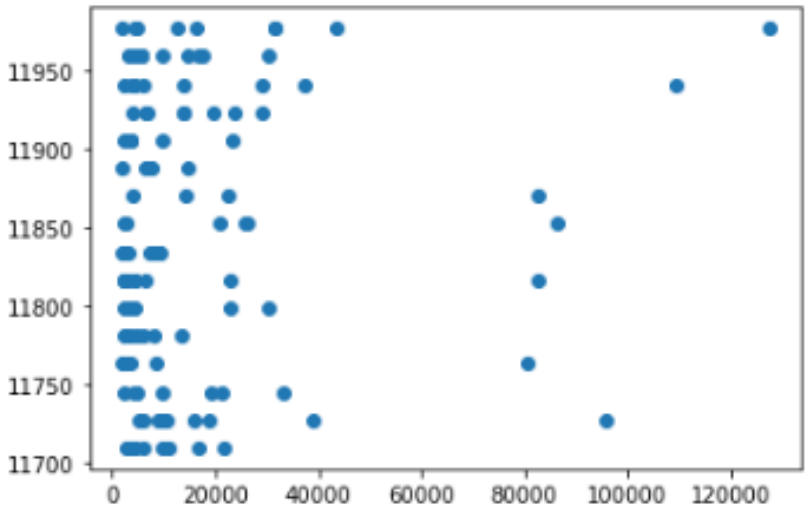
El código empieza por:

1. Tres filtros por si el usuario le gustaría cambiar los parámetros para encontrar la configuración más conveniente.
2. Después se le aplica los filtros al conjunto de datos.
3. Acto seguido se importa las librerías necesarias para utilizar este modelo estadístico, luego se separa la X (variable usada para predecir) y Y (variable usada a predecir).
4. Penúltimo, se divide el conjunto de datos en cuatro para las diferentes fases del modelo de machile Learning, se entrena el modelo y se saca las predicciones.
5. Por último, imprime las métricas del RMSE y el Variance Score.
6. Al final, se grafica visualmente el resultado mediante un scatterplot.

Para esta configuración del modelo, con el "Measure" de "Arrivals", tomando en cuenta los últimos 15 años y un valor mínimo para el "Value" de 2000 personas, obtuvimos un RMSE de 22272.080718992667 y un Variance Score de 0.08%. El RMSE te indica que tan dispersos están tus datos del uno del otro y el Variance Score es un numero entre el -100% al 100% donde te indica que tan correcto son tus datos. En este caso el número es desalentador ya que estas dos variables junto con los filtros aplicados no logran encontrar una correlación.

Ilustración 19 Resultados Linear Regression

```
RMSE:  22272.080718992667
Variance Score:  0.08 %
```



Se confirma que esta configuración no es la óptima para ser utiliza en la predicción de personas que migran a NZ. Se recomiendo utilizar otros parámetros o agregar más variables para un mejor análisis.

Segundo Modelo

A continuación, se realiza otra evaluación del modelo, esta vez agregando todas las opciones en "Citizenship" y manteniendo el filtro por "Year", por el mínimo de "Value" y "Measure". En esta ocasión se mantienen los mismos valores que en el algoritmo pasado para notar alguna mejora.

Ilustración 20 Código Linear Regression con Citizenship

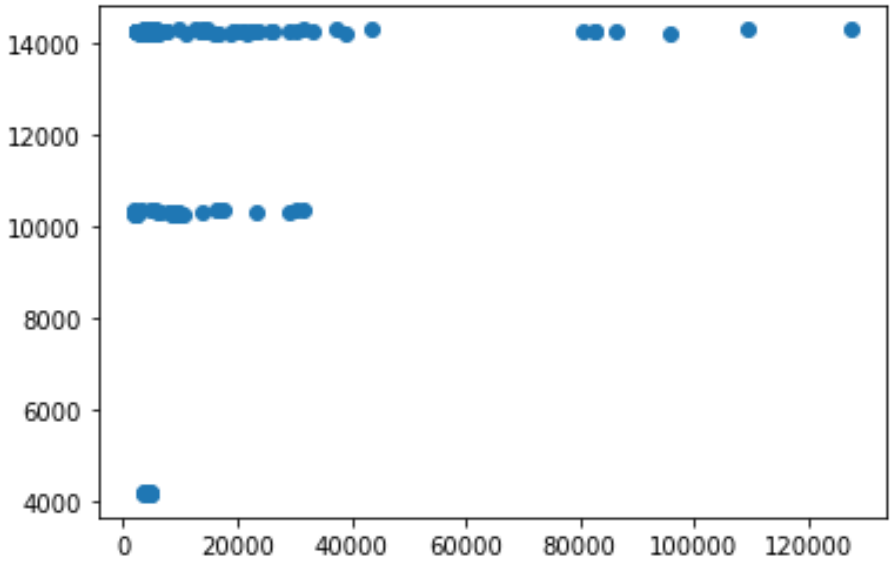
Creación de un Linear Regression por año y nacionalidad para predecir la cantidad de personas con un filtro de metodo de viaje. Con la metrica del RMSE y el Variance Score

```
measure_eleccion = "Arrivals" # Arrivals Departures Net
en_los_ultimos = 15 # Si no quieres filtrar por años, poner 38
thresholder_value = 2000

copy_mod = df_with_dummies_1.copy()
copy_mod_1 = copy_mod.where(copy_mod["Measure"] == measure_eleccion).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Year"] >= max(copy_mod_1["Year"]-en_los_ultimos)).dropna()
copy_mod_3 = copy_mod_2.where(copy_mod_2["Value"] >= thresholder_value).dropna()

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import mean_squared_error

X_lm = copy_mod_3[["Year", "Total All Citizenships", "New Zealand Citizen"]]
y_lm = copy_mod_3["Value"]

X_train_lm, X_test_lm, y_train_lm, y_test_lm = train_test_split(X_lm, y_lm, test_size=0.30, random_state = 101)

lm = LinearRegression().fit(X_train_lm,y_train_lm)
lm_pred = lm.predict(X_test_lm)

print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test_lm, lm_pred)))
print("Variance Score: ", round(100*(metrics.explained_variance_score(y_test_lm, lm_pred)),2),"%")

plt.scatter(y_test_lm, lm_pred)
plt.show()
```

El código de esta versión es el mismo, solo que en la X se le agregan dos variables dummies más, hay una tercera opción, pero se omitió por el hecho que puede ser explicada por esas dos variables cuando el resultado de ambas sean cero.

Ilustración 21 Código Linear Regression con Citizenship

```
RMSE:  21778.75546193285
Variance Score:  3.69 %
```



Para esta configuración del modelo, con el "Measure" de Arrivals, tomando en cuenta los últimos 15 años y un valor mínimo para el "Value" de 2000 personas, obtuvimos un RMSE de 21778.75546193285 y un Variance Score de 3.69 %. En este caso el número mayor que el anterior pero aun es muy débil para explicar una correlación fuerte. Por ende, no se recomienda tomarlo en cuenta para predecir un modelo migratorio.

Tercer Modelo

Otra recomendación seria agregar más variables dependientes para ver si de alguna manera podrían explicar la variable a predecir, en este caso se crearon otras dummies variables en la columna de "Country" donde se extrajo el top 5 de países más populares y la columna de "Measure".

Ilustración 22 Código Linear Regression con varias X

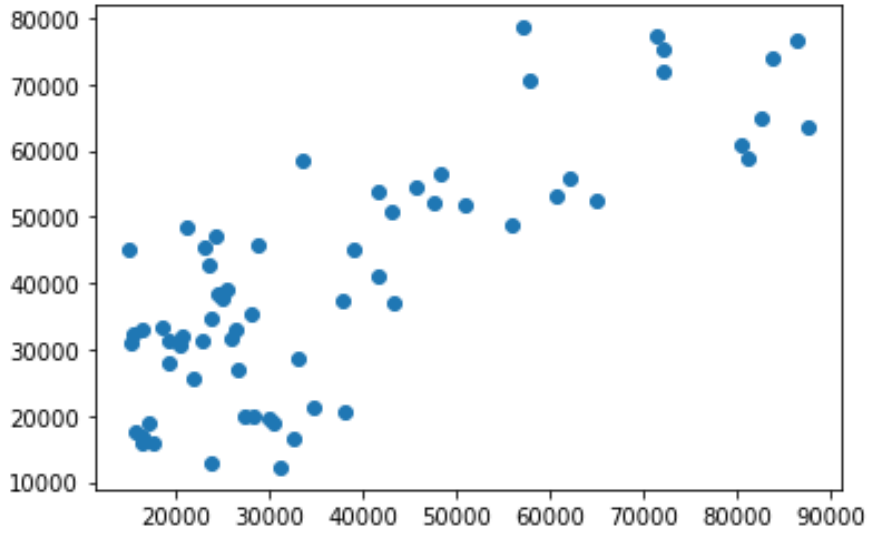
Creación de un Linear Regression predecir la cantidad de personas con un gran conjunto de X. Con la metrica del RMSE y el Variance Score

```
en_los_ultimos = 15 # Si no quieres filtrar por años, poner 38
thresholder_value = 15000

copy_mod = df_with_dummies_3.copy()
copy_mod_1 = copy_mod.where(copy_mod["Year"] >= max(copy_mod["Year"])-en_los_ultimos).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Value"] >= thresholder_value).dropna()

X_lm = copy_mod_2[["Year","New Zealand Citizen", "Total All Citizenships","Departures",'Net',
       'Asia', 'Australia', 'Europe', 'Oceania']]
y_lm = copy_mod_2["Value"]

X_train_lm, X_test_lm, y_train_lm, y_test_lm = train_test_split(X_lm, y_lm, test_size=0.30, random_state = 101)

lm = LinearRegression().fit(X_train_lm,y_train_lm)
lm_pred = lm.predict(X_test_lm)

print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test_lm, lm_pred)))
print("Variance Score: ", round(100*(metrics.explained_variance_score(y_test_lm, lm_pred)),2),"%")

plt.scatter(y_test_lm, lm_pred)
plt.show()
```

El código es el mismo que en anteriores modelos, solo que se agregan las variables categóricas en X de las columnas “Citizenships”, “Measure” y el top 5 de “Country”. Se mantienen los filtros por “Year” y “Value”.

Ilustración 23 Resultados Linear Regression con varias X

```
RMSE:  13172.72440451143
Variance Score:  62.45 %
```



En esta situación, tomando los últimos 15 años y con un mínimo de 15 000 personas, se mejoró considerablemente tanto el RMSE como el Variance Score, bajo estos parámetros y las variables dependiente se puede predecir Y en un 62%, si bien es cierto no es tan bueno este número, se acerca más a un modelo para ser utilizado.

Cuarto Modelo

Como último modelo de regresión lineal, se tiene pensado estandarizar los datos en la columna "Value", esto ya que los datos están bastante distantes, por ende, se busca acortar la distancia mediante transformar los datos bajo una misma escala para verificar si esto podría afectar al rendimiento del modelo de Machine Learning.

Ilustración 24 Estandarización de datos

Haciendo una estandarizacion de datos para notar si hay una mejora

```
min_value = df_with_dummies_3["Value"].min()
max_value = df_with_dummies_3["Value"].max()

df_with_dummies_3_Std = df_with_dummies_3.copy()

df_with_dummies_3_Std["Value_Normalized"] = (df_with_dummies_3_Std["Value"] - min_value) / (max_value - min_value)

df_with_dummies_3_Std.head()
```

Se mantiene el código tal cual, solo que en la Y cambiaria por la columna estandarizada.

Ilustración 25 Código Linear Regression con múltiples X estandarizado

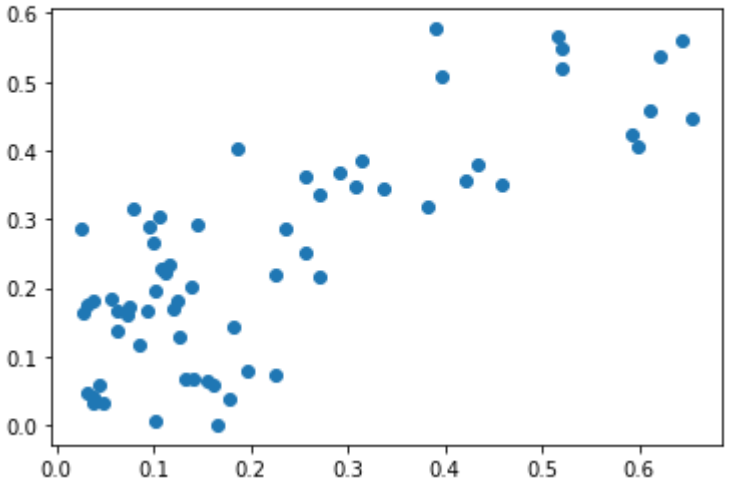
Creación de un Linear Regression predecir la cantidad de personas con un gran conjunto de X estandarizado. Con la metrica del RMSE y el Variance Score

```
en_los_ultimos = 15 # Si no quieres filtrar por años, poner 38
thresholder_value = 15000

copy_mod = df_with_dummies_3_Std.copy()
copy_mod_1 = copy_mod.where(copy_mod["Year"] >= max(copy_mod["Year"])-en_los_ultimos).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Value"] >= thresholder_value).dropna()

X_lm = copy_mod_2[["Year","New Zealand Citizen", "Total All Citizenships","Departures",'Net',
       'Asia', 'Australia', 'Europe', 'Oceania']]
y_lm = copy_mod_2["Value_Normalized"]

X_train_lm, X_test_lm, y_train_lm, y_test_lm = train_test_split(X_lm, y_lm, test_size=0.30, random_state = 101)

lm = LinearRegression().fit(X_train_lm,y_train_lm)
lm_pred = lm.predict(X_test_lm)

print("RMSE: ", np.sqrt(metrics.mean_squared_error(y_test_lm, lm_pred)))
print("Variance Score: ", round(100*(metrics.explained_variance_score(y_test_lm, lm_pred)),2),"%")

plt.scatter(y_test_lm, lm_pred)
plt.show()
```

Como resultado, el RMSE bajó, pero es normal porque los datos están a otra escala menor, pero el Variance score se mantiene, entonces se confirma que estandarizar la Y no fue de gran impacto para el resultado del algoritmo.

Ilustración 26 Resultado Linear Regression con múltiples X estandarizado

```
RMSE:  0.11434855122928773
Variance Score:  62.45 %
```



Regresión Logística

Este modelo de Machine Learning es útil cuando se precisa predecir variables categóricas, viendo el conjunto de datos, se nota que tenemos tres columnas categóricas, que serían: “Citizenship”, “Country” y “Measure”. Se podría hacer una lógica inversa con respecto a los anteriores modelos de regresión linear, usar el “value” para predecir y no para predecirlo.

Primer modelo

Como base en este primer algoritmo, se intentó predecir el “Citizenship” con base al número de personas. Se mantienen los tres filtros originales, en los últimos años, un mínimo de personas y por tipo de “Measure”, esto para contemplar todos los escenarios y buscar el patrón correcto a utilizar para maximizar los resultados.

Ilustración 27 Código regresión logística con filtros

Creación de un Logistic Regression cantidad de personas tratando de predecir la nacionalidad. Con métricas de Precision, recall y F1 y con una tabla de confución. ¶

```
measure_eleccion = "Net" # Arrivals Departures Net
en_los_ultimos = 10 # Si no quieres filtrar por años, poner 38
thresholder_value = 500

copy_mod = df_with_dummies_1.copy()
copy_mod_1 = copy_mod.where(copy_mod["Measure"] == measure_eleccion).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Year"] >= max(copy_mod_1["Year"]-en_los_ultimos)).dropna()
copy_mod_3 = copy_mod_2.where(copy_mod_2["Value"] >= thresholder_value).dropna()

X = copy_mod_3[['Value']]
y = copy_mod_3["Citizenship"]

X = X.replace([np.inf, -np.inf], np.nan).fillna(0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state = 101)

from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression().fit(X_train, y_train)
pre = logmodel.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print("Classification Report: ")
print(classification_report(y_test, pre))
print("********************************")
print("Confusion Matrix: ")
print(confusion_matrix(y_test, pre))
```

El código empieza por:

1. Tres filtros por si el usuario le gustaría cambiar los parámetros para encontrar la configuración más conveniente.
2. Después se le aplica los filtros al conjunto de datos.
3. Luego se separa la X (variable usada para predecir) y Y (variable usada a predecir).
4. Se elimina cualquier tipo de impureza en los datos como infinitos y NaN
5. Se divide el conjunto de datos en cuatro para las diferentes fases del modelo de machile Learning, se entrena el modelo y se saca las predicciones.
6. Se imprime el reporte de clasificación y la matriz de confusión.

Antes de mostrar los resultados, primero hay que explicar dos conceptos. El reporte de clasificación y la matriz de confusión.

El reporte de clasificación es un modelo de entrega de datos donde se presenta los resultados del modelo de clasificación mayoritariamente, en este, nos muestra todas las variables posibles (en este caso son tres, Australian, New Zealand y Total All Citizenships). Juntos a estas, nos muestran el nombre de las métricas y su correspondiente número.

Estas son:

Precision: La precisión mide la exactitud de las predicciones positivas del modelo. Es decir, de todas las predicciones que el modelo hizo como positivas, cuántas eran realmente positivas. Y se calcula dividiendo todos los True Positive (TP) entre los TP + False Positive (FP)

Recall: El recall mide la capacidad del modelo para detectar todas las instancias positivas. Es decir, de todas las instancias que son realmente positivas, cuántas el modelo identificó correctamente. Y se calcula dividiendo todos los True Positive (TP) entre los TP + False Negative (FN)

F1-Score: El F1-score es la media armónica entre la precisión y el recall. Se utiliza cuando se quiere encontrar un balance entre estos dos indicadores, especialmente si los datos son desbalanceados. Y se calcula multiplicando por dos la división entre Precision + Recall y Precision x Recall

Support: El soporte es el número de ocurrencias reales de cada clase en el conjunto de datos. En otras palabras, es la cantidad de muestras que pertenecen a cada clase.

Y de ultimo tenemos la matriz de confusión, que es una herramienta que nos facilita ver los TP, TN, FP y FN. En otras palabras, nos permite ver cuantas predicciones fueron correctas e incorrectas. La matriz suele construirse por el número de variables a lo largo y a lo ancho por las predicciones. Una matriz sana es aquella que, de manera diagonal, sus números son los mal altos.

Ilustración 28 Matriz de confusión sana

	Predicción A	Predicción B	Predicción C
Real A	50	2	1
Real B	10	45	5
Real C	0	3	52

Una vez explicado el reporte de clasificación y la matriz de confusión, analicemos los resultados del primer modelo de regresión Logística. Con filtros de Net en "Measure", en los último 10 años y con un mínimo de personas de 500.

Ilustración 29 Resultados regresión logística con filtros

```
Classification Report:
                        precision    recall  f1-score   support

     Australian Citizen       0.00      0.00      0.00         7
   New Zealand Citizen        0.00      0.00      0.00         2
Total All Citizenships        0.90      1.00      0.95        79

              accuracy                           0.90        88
             macro avg        0.30      0.33      0.32        88
          weighted avg        0.81      0.90      0.85        88

**********
Confusion Matrix:
[[ 0  0  7]
 [ 0  0  2]
 [ 0  0 79]]
```

Si bien es cierto obtuvimos un resultado muy alto de predicción, en la columna de support podemos observar que hay una mayor población en una variable, y es lo suficientemente grande como para que los errores de otras variables no afecten al resultado final. Aunque podemos concluir que el Net en la variable "Net" es muy importante para el análisis.

Segundo Modelo

Este modelo se parece al anterior. Solo que en X se agrega la variable de Measure en forma de dummy variables y la variable de "Year", tiene como fin confirmar si segmentando más el entrenamiento, el resultado de predicción pueda ser mayor y de rebote, si la variable Measure es significativa para otros modelos de machine Learning.

Ilustración 30 Código Regresión Linean con Measure y Year

Creación de un Logistic Regression cantidad de personas y el método de viaje tratando de predecir la nacionalidad. Con métricas de Precision, recall y F1 y con una tabla de confución.

```
en_los_ultimos = 10 # Si no quieres filtrar por años, poner 38
thresholder_value = 500

copy_mod = df_with_dummies_2.copy()
copy_mod_1 = copy_mod.where(copy_mod["Year"] >= max(copy_mod_1["Year"]-en_los_ultimos)).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Value"] >= thresholder_value).dropna()

X = copy_mod_2[['Value', "Year", "Departures", "Net"]]
y = copy_mod_2["Citizenship"]

X = X.replace([np.inf, -np.inf], np.nan).fillna(0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state = 101)

from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression().fit(X_train, y_train)
pre = logmodel.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print("Classification Report: ")
print(classification_report(y_test, pre))
print("********************************")
print("Confusion Matrix: ")
print(confusion_matrix(y_test, pre))
```

Ilustración 31 Resultado Regresión Linean con Measure y Year

```
Classification Report:
                       precision    recall  f1-score   support

    Australian Citizen       0.00      0.00      0.00        28
   New Zealand Citizen       0.00      0.00      0.00        69
Total All Citizenships       0.73      1.00      0.84       262

              accuracy                           0.73       359
             macro avg       0.24      0.33      0.28       359
          weighted avg       0.53      0.73      0.62       359

********************************
Confusion Matrix:
[[  0   0  28]
 [  0   0  69]
 [  0   0 262]]
```

El resultado no fue el esperado, se creía que iba a ser más alto o evitar el problema del modelo pasado, donde el resultado de una variable afecta al performance porque hay poca de las otras opciones. Pero se concluye que “Total All Citizenships” tiene una alta presencia en el dataset para futuros modelos.

Tercer Modelo

Con el afán de incluir más variables al modelo para notar de alguna manera un patrón entre columnas, ahora se agrega para el entrenamiento los top 5 países más frecuentes, se considera esta variable importante ya que por la cercanía que hay entre países, es un factor importante para determinar el flujo de migración.

```
en_los_ultimos = 10 # Si no quieres filtrar por años, poner 38
thresholder_value = 500

copy_mod = df_with_dummies_3.copy()
copy_mod_1 = copy_mod.where(copy_mod["Year"] >= max(copy_mod_1["Year"]-en_los_ultimos)).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Value"] >= thresholder_value).dropna()

X = copy_mod_2[['Value', "Year", "Departures", "Net", 'Asia', 'Australia', 'Europe', 'Oceania']]
y = copy_mod_2["Citizenship"]

X = X.replace([np.inf, -np.inf], np.nan).fillna(0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state = 101)

from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression().fit(X_train, y_train)
pre = logmodel.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print("Classification Report: ")
print(classification_report(y_test, pre))
print("******************************")
print("Confusion Matrix: ")
print(confusion_matrix(y_test, pre))
```

```
Classification Report:
                         precision    recall  f1-score   support

   New Zealand Citizen       0.31      0.56      0.40         9
Total All Citizenships       0.89      0.74      0.81        43

              accuracy                           0.71        52
             macro avg       0.60      0.65      0.61        52
          weighted avg       0.79      0.71      0.74        52

********************************
Confusion Matrix:
[[ 5  4]
 [11 32]]
```

Analizando los resultados, se nota dos cambios, el primero es que el modelo está defiriendo a más de una variable y la segunda es que según con las variables dependientes, una opción (Australian Citizen) no se está tomando en cuenta, podría ser que esta misma sea el determinante para que en los modelos anteriores el modelo no distinga entre las tres opciones.

Random Forest

Como se mencionaron anteriormente en los dos modelos pasados, la regresión lineal y logísticas cuentan con diferencias esenciales para usarlos, uno de esas diferencias es el resultado. La regresión lineal es exclusiva para predecir valores continuos y la regresión logística es exclusiva para predecir etiquetas.

El algoritmo de random forest puede trabajar con ambos resultados, tanto predicción de etiquetas como predicción de valores continuos. Solo que se debe de configurar como quieres recibir el resultado. Esto es de suma utilidad ya que tienes un modelo todo terreno que te ayuda a la experimentación.

El modelo de Random Forest se basa en múltiples arboles de decisión, estos mismos son como caminos que toma el modelo para predecir un valor o etiqueta. Random Forest crea n cantidad de árboles de decisión que se entrenó cada uno diferente, y promedia sus resultados o saca la moda. Este enfoque ayuda a capturar una variedad de patrones en los datos y proporciona una mayor robustez frente a las anomalías y el ruido.

Primer modelo

Para esta primera experimentación, se plantea un algoritmo con filtros en países, que fueron elegidos principalmente por su cercanía al país de Nueva Zelanda, también filtros por años y por mínimo de personas. Se logra buscar si se puede predecir la cantidad de personas dependiendo de cuál sea su país de procedencia.

Ilustración 32 Codigo Random Forest solo una X

Creación de un Random Forest con la cantidad de personas tratando de predecir el país en un n país. Con métricas de Precision, recall y F1 y con una tabla de confución.

```
paises_elegir = ["Oceania", "Australia"]
en_los_ultimos = 1 # Si no quieres filtrar por años, poner 38
thresholder_value = 5000
n_estimators_preferred = 1000

column_names = ['Measure', 'Country', 'Citizenship', 'Year', 'Value',
       'New Zealand Citizen', 'Total All Citizenships', 'Departures', 'Net']


df_mod_rf = pd.DataFrame(columns=column_names)

for i in paises_elegir:
    df_filter = df_with_dummies_2.where(df_with_dummies_2["Country"] == i).dropna()
    df_mod_rf = pd.concat([df_mod_rf, df_filter])


copy_mod = df_mod_rf.copy()
copy_mod_1 = copy_mod.where(copy_mod["Year"] >= max(copy_mod["Year"])-en_los_ultimos).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Value"] >= thresholder_value).dropna()

X = copy_mod_2[["Value"]]
y = copy_mod_2["Country"]

X_train, X_test, y_train, y_test = train_test_split( X,y, test_size=0.3)

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=n_estimators_preferred)
rfc.fit(X_train, y_train)

pre = rfc.predict(X_test)

print(confusion_matrix(y_test, pre))
print("********************")
print(classification_report(y_test, pre))
```

El código empieza por:

1. Cuatro filtros por si el usuario le gustaría cambiar los parámetros para encontrar la configuración más conveniente.
2. Se crea el esqueleto para un conjunto de datos temporal
3. Se extrae, mediante un for, la información del dataset original dependiendo del país para al final hacer uno solo.
4. Después se le aplica los filtros al conjunto de datos.
5. Se divide el data set en X (variable usada para predecir) y Y (variable usada a predecir).

6. Se divide el conjunto de datos en cuatro para las diferentes fases del modelo de machile Learning, se entrena el modelo y se saca las predicciones.
7. Acto seguido se importa las librerías necesarias para utilizar este modelo estadístico.
8. Penúltimo, se entrena el modelo para después generar predicciones.
9. Se imprime el reporte de clasificación y la matriz de confusión.

Ilustración 33 Resultados Random Forest solo una X

```
[[4 0]
 [0 2]]
********************
              precision    recall  f1-score   support

    Australia       1.00      1.00      1.00         4
      Oceania       1.00      1.00      1.00         2

     accuracy                           1.00         6
    macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6
```

Ya explicado anteriormente como se conforma una matriz de confusión y el reporte de clasificación, podemos observar una precisión perfecta en la predicción, aunque esto no es del todo bueno, ya que, según nuestros filtros, de la alguna manera estamos obligando al modelo a obtener ese resultado.

Primero por el filtro de los países, ya que estos están altamente cercanos al NZ, esto afecta ya que las personas están más atenuantes a migrar a dicho país, también se tomó como referencia años a un corto periodo, lo cual, en circunstancias normales, pues será fácil de predecir. También con un n_estimators de 1000, se le está dando oportunidad al modelo de ser más estable con su resultado.

Segundo Modelo

En el anterior modelo de Random Forest, solo se utilizó una X para predecir y, en esta ocasión se va agregar más variables dependientes para concluir si introducir más información va ser más beneficioso para el modelo. Los filtros se mantienen igual para tener la certeza que solo se esté afectando por la cantidad de X.

Ilustración 34 Código Random Forest varias X

```
paises_elegir = ["Oceania", "Australia"]
en_los_ultimos = 1 # Si no quieres filtrar por años, poner 38
thresholder_value = 5000
n_estimators_preferred = 1000

column_names = ['Measure', 'Country', 'Citizenship', 'Year', 'Value',
       'New Zealand Citizen', 'Total All Citizenships', 'Departures', 'Net']


df_mod_rf = pd.DataFrame(columns=column_names)

for i in paises_elegir:
    df_filter = df_with_dummies_2.where(df_with_dummies_2["Country"] == i).dropna()
    df_mod_rf = pd.concat([df_mod_rf, df_filter])


copy_mod = df_mod_rf.copy()
copy_mod_1 = copy_mod.where(copy_mod["Year"] >= max(copy_mod["Year"])-en_los_ultimos).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Value"] >= thresholder_value).dropna()

X = copy_mod_2[["Value", 'Year',
       'Departures', 'Net']]
y = copy_mod_2["Country"]

X_train, X_test, y_train, y_test = train_test_split( X,y, test_size=0.3)

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=1000)
rfc.fit(X_train, y_train)

pre = rfc.predict(X_test)

print(confusion_matrix(y_test, pre))
print("********************")
print(classification_report(y_test, pre))
```

Ilustración 35 Resultados Random Forest varias X

```
[[1 0]
 [2 3]]
********************
              precision    recall  f1-score   support

   Australia       0.33      1.00      0.50         1
     Oceania       1.00      0.60      0.75         5

    accuracy                           0.67         6
   macro avg       0.67      0.80      0.62         6
weighted avg       0.89      0.67      0.71         6
```

Como se puede observar, el agregar más variables para tratar de predecir con más precisión no es viable en esta situación, esto puede ser porque hay variables que hacen ruido o confunden al modelo y por ende tomar decisiones equivocadas. Se concluye que lo mejor sería investigar acerca de las variables que podrían estar ocasionando confusión al modelo.

PCA

El PCA (Principal Components Analysis) o en español, análisis principal de componentes es una técnica estadística que se utiliza para reducir la dimensionalidad de un conjunto de datos. Esto es importante ya que nos ayuda a identificar cuáles son las variables más significativas para el entrenamiento de un modelo de Machine Learning.

Primer Modelo

Se desea saber cuáles con los componentes principales en el dataset que se ha utilizado en todo el documento, se incluyen todas las variables del conjunto de datos, tanto numéricas continuas, enteras y booleanas. Se espera obtener las columnas más importantes que influyen en el modelo para su nivel de predicción.

Ilustración 36 Código PCA

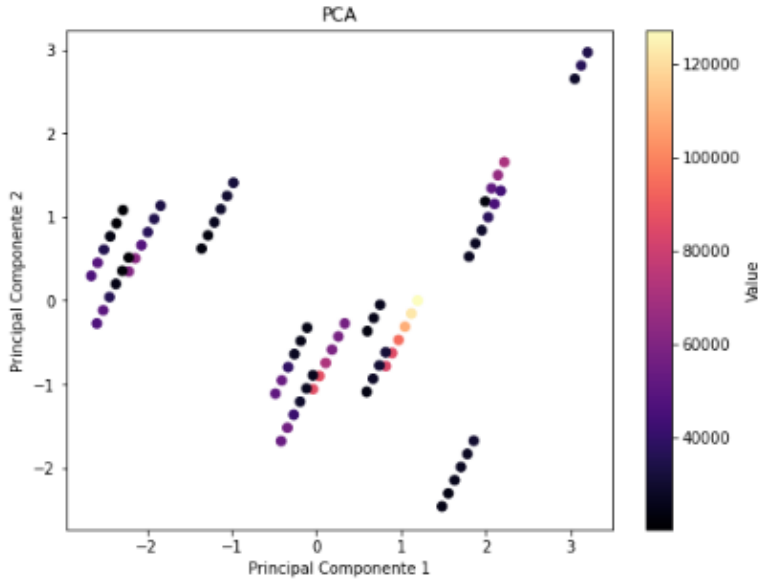
Creación de un PCA con el conjunto de datos

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

en_los_ultimos = 5 # Si no quieres filtrar por años, poner 38
thresholder_value = 20000

copy_mod = df_with_dummies_3.copy()
copy_mod_1 = copy_mod.where(copy_mod["Year"] >= max(copy_mod["Year"])-en_los_ultimos).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Value"] >= thresholder_value).dropna()

X = copy_mod_2[['Year','New Zealand Citizen', 'Total All Citizenships', 'Departures', 'Net',
       'Asia', 'Australia', 'Europe', 'Oceania']]


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)


plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=copy_mod_2['Value'], cmap='magma')
plt.xlabel('Principal Componente 1')
plt.ylabel('Principal Componente 2')
plt.title('PCA')
plt.colorbar(label='Value')
plt.show()


print('Los variance score son: ', pca.explained_variance_ratio_)

first_pc = pca.components_[0]
second_pc = pca.components_[1]


print('Primer componente principal :', first_pc)
print('Segundo componente principal :', second_pc)
```

El código empieza por:

1. Importe de las librerías necesarias.
2. Filtro por año y por cantidad mínima de persona.
3. Se aplican los filtros y se crea X con todas las columnas necesarias.
4. Se estandariza los valores de X.
5. Se entrena el PCA con dos componentes ya que es lo ideal.
6. Se crea un scatterplot donde refleje, según el "Value", cuáles son los principales componentes para el dataset.
7. Se imprime el Variance Score y el array de ambos componentes.

Ilustración 37 Resultado de PCA




```
Los variance score son:  [0.30505049 0.14931017]
************************************************************
Primer componente principal : [ 0.12530861 -0.50895209  0.50895209 -0.43007346  0.29942823  0.32107574
 -0.1783873   0.1806893  -0.15508888]
Segundo componente principal : [ 0.26291989  0.32955048 -0.32955048 -0.13733654  0.48484348  0.42876915
 -0.02106376 -0.45833113 -0.25569019]
```

El variance score o variance ratio es el nivel representativo de los componentes principales, es decir, que tanto pueden explicar todo el conjunto de datos. En este ejemplo obtenemos que el componente número uno explica el 30%y el segundo el 15%, si sumamos los dos, podemos concluir que ambos componentes pueden predecir el 45% del dataset.

Seguido tenemos el vector de cada componente. El algo del vector dependerá de cuantas variables se colocan en X, en este caso se entrenó con nueve variables, por ende, el largo del array será nueve. Dentro de él, se encuentran los números en que cada variable representa un porcentaje del variance score final.

Como se mencionó anteriormente, los resultados de los dos componentes fueron de 0.3 y 0.15, dando como total una varianza del 0.45. Este número es muy bajo, de hecho, confirma que no hay una variable totalmente crucial para la predicción de la columna "Value". Solo hay algunas variables que tienen más impacto que otras, pero no a un nivel significativo.

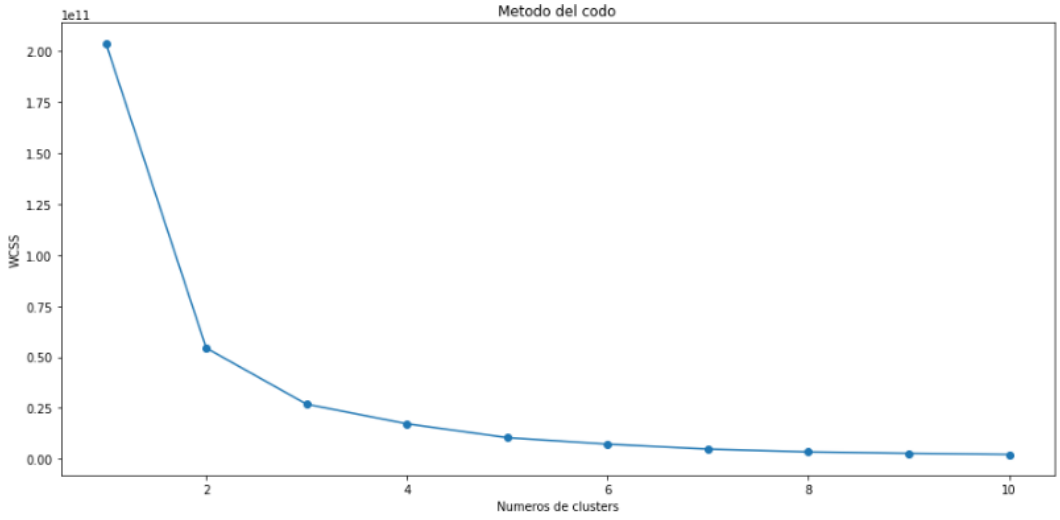
Analizando los vectores de los componentes principales, se puede analizar que por su alto valor entre todas. En el componente uno, las variables más significativas serian: New Zealand Citizen, Total All Citizenships y Departures. Y en el componente dos, las variables más significativas serian: Net, Asia y Europe. Todas estas variables explican el 0.45 del dataset.

K-MEANS

El modelo de K-MEANS es un algoritmo tipo clustering, este tipo de algoritmo trata de clasificar todos los datos en un plano cartesiano, donde dependiendo de sus características, van a permanecer a un grupo o al otro, depende de cuantos grupos desea crear. Para saber lo anterior, existe una técnica que permite saber cuántos grupos serían los óptimos a utilizar.

Primer Modelo

Como se mencionó anteriormente, antes de empezar con el algoritmo de KMEANS, primero necesitamos saber cuál es la cantidad óptimo de grupos, para ese necesitamos un gráfico de codo, esta técnica testea desde un rango de números el mejor número, esto lo sabe mediante el WSS, y donde se empieza a estabilizar, ese sería el número idea.



En este caso, se arrojó el número cuatro, por ende, dentro de la configuración del modelo de machine learning, el número de grupos a elegir seria este dicho número.

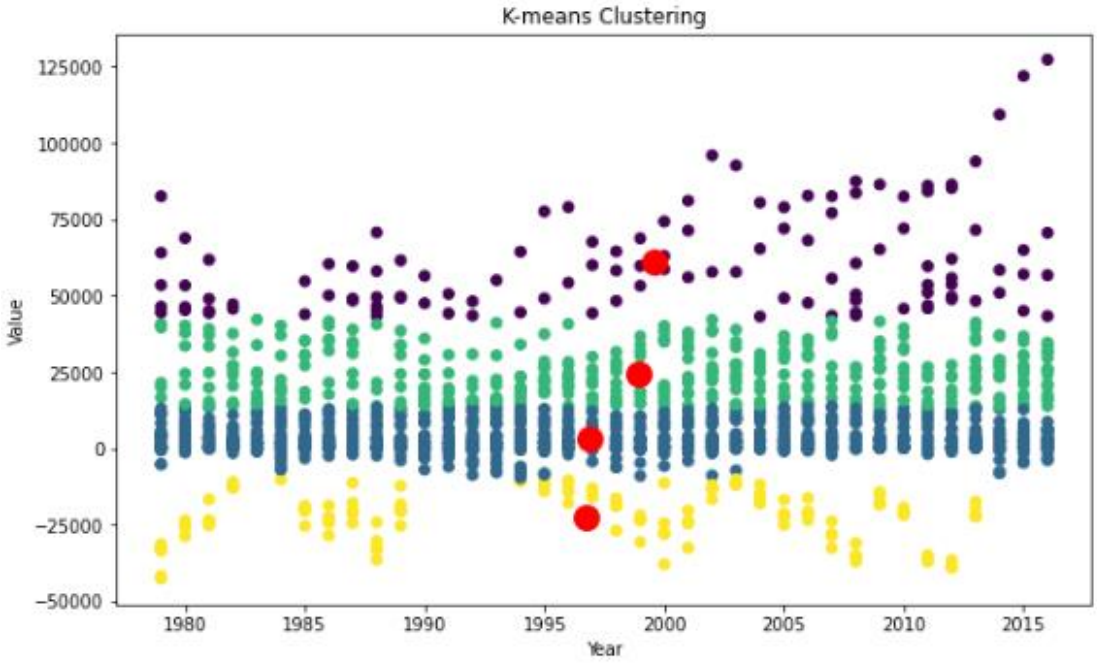
Ilustración 38 Código de Kmeans

```
en_los_ultimos = 5 # Si no quieres filtrar por años, poner 38
thresholder_value = 2000

copy_mod = df_with_dummies_3.copy()
copy_mod_1 = copy_mod.where(copy_mod["Year"] >= max(copy_mod["Year"])-en_los_ultimos).dropna()
copy_mod_2 = copy_mod_1.where(copy_mod_1["Value"] >= thresholder_value).dropna()

numeric_df = df_with_dummies_3[['Year', 'Value',
       'New Zealand Citizen', 'Total All Citizenships', 'Departures', 'Net']]

numeric_df.dropna()

kmeans = KMeans(n_clusters=4)
kmeans.fit(numeric_df)
labels = kmeans.labels_
centers = kmeans.cluster_centers_

numeric_df.dropna()

plt.figure(figsize=(10, 6))
plt.scatter(numeric_df['Year'], numeric_df['Value'], c=labels, cmap='viridis')
plt.scatter(centers[:, 0], centers[:, 1], marker='o', s=200, color='red')
plt.xlabel('Year')
plt.ylabel('Value')
plt.title('K-means Clustering')
plt.show()
```

El código empieza por:

1. Se colocan los filtros para personalizar el entrenamiento del modelo.
2. Se aplican los filtros al conjunto de datos.
3. Se crea un dataset exclusivo con variables numéricas y se eliminan los nulls.
4. Se entrena el modelo con el número de clústeres recomendado.
5. Se extraen las etiquetas y los centros.
6. Se imprime un scatterplot destacando de color los grupos y sus centros.

Ilustración 39 Resultado K means



Como resultado tenemos un scatterplot con los cuatro grupos bien distinguidos y con color rojo sus centros, a simple vista se nota que el modelo separado los datos por su valor y no por años, esto ayuda a comprender que hay datos que se comportan igual, como siguiente paso podríamos ver por separado dichos grupos para aplicar otra técnica de machine learning.

TIPO DE DATA MINING

Dentro del Data Mining hay diferentes tipos de especializaciones, una de esas técnicas es la predictiva, que se enfoca en predecir resultados futuros basados en patrones y tendencias. Se basa en algoritmos estadísticos y de aprendizaje automático para identificar relaciones y patrones en los datos que pueden utilizarse para hacer predicciones sobre eventos futuros.

Dado a las características del conjunto de datos a trabajar, junto con los posibles problemas que podría solucionar, el data mining predictivo es el que le podemos sacar más provecho por sus atributos para pronosticar eventos futuros con datos del pasado. Con el fin de tomar acciones más precisas y que nos brinden del mayor beneficio posible.

Gracias a las técnicas de la minería de datos predictivas podrías predecir futuros niveles de oferta y demanda basada al aumento o disminución de personas. A la vez, crear presupuestos quinquenales para generar estabilidad económica mediante políticas monetaria y fiscal acorde a las necesidades del momento.

CONCLUSIONES

- Satisfactoriamente se obtuvieron varias medidas estadísticas que lograron brindar más información sobre el conjunto de datos, esto siendo la base de otras futuras etapas para un análisis predictivo más formal.
- Se obtuvo un análisis profundo sobre los patrones y tendencias del tema a tratar mediante la ejecución de gráficos representativos donde intuitivamente se ejemplifican dichas características.
- Se implementaron técnicas de manipulación de datos, incluyendo la sustitución de valores y filtros, con el objetivo de extraer el máximo conocimiento posible del conjunto de datos, estableciendo así las bases sólidas para el análisis subsiguiente.

RECOMENDACIONES

- Alentamos al lector a investigar a profundo la rama del Data Mining y todos sus beneficios, esto ya que es una herramienta sumamente efectiva para la investigación y resolución de problemas.
- Encarecidamente incentivamos a todos los organismos, ya sean públicos o privados, a la inversión para la implementación de Data Mining dentro de ellas, esto ya que está más que demostrado la efectivad para la toma de decisiones de alto impacto.

BIBLIOGRAFÍA

Amazon. (s.f.). *AWS*. Obtenido de AWS: https://aws.amazon.com/es/what-is/data-analytics/#:~:text=El%20an%C3%A1lisis%20de%20datos%20puede,mejorar%20exponencialmente%20el%20rendimiento%20empresarial.

Leyva, J. L. (24 de 05 de 2023). *Universidad Veracruzana*. Obtenido de Universidad Veracruzana: https://www.uv.mx/prensa/general/destacan-papel-de-la-estadistica-en-la-toma-de-decisiones-sociales/

ONU. (21 de 9 de 2016). *ONU*. Obtenido de ONU: https://refugeesmigrants.un.org/es/la-migraci%C3%B3n-es-beneficiosa-para-todos-si-se-gestiona-correctamente#:~:text=La%20migraci%C3%B3n%20puede%20propiciar%20un,para%20los%20pa%C3%ADses%20de%20origen.