Tipología y ciclo de vida de los datos

Práctica 2

Alicia Amores y Carlos Núñez

1. Descripción del dataset

¿Por qué es importante y qué pregunta/problema pretende responder?

El dataset contiene información de diferentes puestos de trabajo relacionados con la ciencia de datos y los salarios mínimos, medios y máximos de cada uno de ellos. En función de estas características se quiere ver qué puesto obtiene mejores salarios y se quiere predecir los salarios máximos para cada tipo de puesto en función de las características descritas.

Algunas características del puesto de trabajo pueden ser: la compañía, el tamaño de la compañía, la localización del puesto, los conocimientos del candidato...

2. Integración y selección de los datos de interés a analizar.

Puede ser el resultado de adicionar diferentes datasets o una subselección útil de los datos originales, en base al objetivo que se quiera conseguir.

Analizamos los campos categóricos

```
data.describe(include='object').T
```

	count	unique	top	freq
Job Title	742	264	Data Scientist	131
Salary Estimate	742	416	$49K-$113K (Glassdoor est.)	6
Job Description	742	463	Description\nMedical Laboratory Scientist - Te...	4
Company Name	742	343	MassMutual\n3.6	14
Location	742	200	New York, NY	55
Headquarters	742	198	New York, NY	52
Size	742	9	1001 to 5000 employees	150
Type of ownership	742	11	Company - Private	410
Industry	742	60	Biotech & Pharmaceuticals	112
Sector	742	25	Information Technology	180
Revenue	742	14	Unknown / Non-Applicable	203
Competitors	742	128	-1	460
company_txt	742	343	MassMutual	14
job_state	742	37	CA	152
job_simp	742	7	data scientist	279
seniority	742	3	na	520

En la columna seniority la mayoría de los valores son na. No se han reconocido como valores nulos, sino como string, pero realmente son valores vacíos que no aportan ninguna información. Por ello se ha decidido eliminar dicha columna.

Por otro lado, la mayoría de los valores del campo Competitors son -1, lo cual no tiene mucho sentido, por ello también se va a eliminar dicha columna.

La columna de Salary Estimate lo podemos determinar en función del salario medio.

Por último también vamos a eliminar la columna de job description ya que nos vamos a enfocar solamente en los roles de Data Scientist y Data Engineer en el modelo estadístico.

```
[45] data.drop(['seniority', 'Competitors', 'Salary Estimate', 'Job Description'], inplace=True, axis=1)
     data.head()
```

	Job Title	Rating	Company Name	Location	Headquarters	Size	Founded	Type of ownership	Industry	Sector	...	same_state	ag
0	Data Scientist	3.8	Tecolote Research\n3.8	Albuquerque, NM	Goleta, CA	501 to 1000 employees	1973	Company - Private	Aerospace & Defense	Aerospace & Defense	...	0	4
1	Healthcare Data Scientist	3.4	University of Maryland Medical System\n3.4	Linthicum, MD	Baltimore, MD	10000+ employees	1984	Other Organization	Health Care Services & Hospitals	Health Care	...	0	3
2	Data Scientist	4.8	KnowBe4\n4.8	Clearwater, FL	Clearwater, FL	501 to 1000 employees	2010	Company - Private	Security Services	Business Services	...	1	1
3	Data Scientist	3.8	PNNL\n3.8	Richland, WA	Richland, WA	1001 to 5000 employees	1965	Government	Energy	Oil, Gas, Energy & Utilities	...	1	5
4	Data Scientist	2.9	Affinity Solutions\n2.9	New York, NY	New York, NY	51 to 200 employees	1998	Company - Private	Advertising & Marketing	Business Services	...	1	2

5 rows × 28 columns

3. Limpieza de datos

- ¿Los datos contienen ceros o elementos vacíos? Gestiona cada uno de estos casos.

  No existen datos faltantes. En la siguiente figura se puede observar de forma visual. En caso de haber espacios en blanco, estos indicarían donde están los datos faltantes.

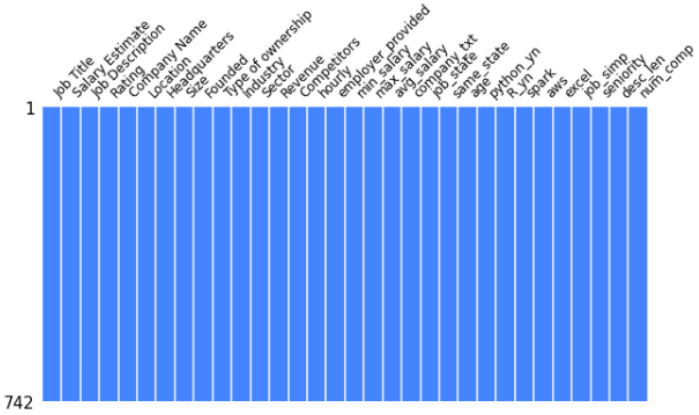
Datos Faltantes

```
# Lista con los missing de cada columna
missing = data.isna().sum()
# Creamos dataframe con columnas y missing values
missing_df = pd.DataFrame([data.columns,missing]).transpose()
# Nos quedamos solo con  las columnas que tengan missing
for index in missing_df.index:
    if missing_df.iloc[index,1] > 0:
        print('Columna: {}, N of NaN: {}, Index in Dataframe: {}'.format(missing_df.iloc[index,0],missing_df.iloc[index,1],index))
```

No existen datos faltantes. En la siguiente figura se puede observar de forma visual. En caso de haber espacios en blanco, estos indicarían donde están los datos faltantes.

```
[28] missingno.matrix(data,sparkline=False, figsize=(10,5), fontsize=12, color=(0.27, 0.52, 1.0))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06f1fb8580>
```



- Identifica y gestiona los valores extremos.

  Al analizar los datos numéricos encontramos algunas columnas que contienen valores outliers (-1) y no tiene sentido por la naturaleza de dicho atributo.

  Las columnas rating, age y founded no tiene sentido que tengan valor -1, ya que una es la valoración, entre 0 y 5, age es la edad de la persona que pide el trabajo y founded es el año de fundación de la empresa. Además, encontramos en el atributo age un outlier de 276 años.

```
stats_df = data.describe()
# Obtener y añadir la mediana
median = pd.DataFrame(data.median())
median = median.transpose()
median.rename(index = {0:'median'},inplace = True)
stats_df = stats_df.append(median).transpose() # Hay q igualarlo porq append() devuelve nuevo df
stats_df
```

```
<ipython-input-46-d1f1a0e2c2cc>:3: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
  median = pd.DataFrame(data.median())
```

	count	mean	std	min	25%	50%	75%	max	median
Rating	742.0	3.618868	0.801210	-1.0	3.3	3.7	4.0	5.0	3.7
Founded	742.0	1837.154987	497.183763	-1.0	1939.0	1988.0	2007.0	2019.0	1988.0
hourly	742.0	0.032345	0.177034	0.0	0.0	0.0	0.0	1.0	0.0
employer_provided	742.0	0.022911	0.149721	0.0	0.0	0.0	0.0	1.0	0.0
min_salary	742.0	74.719677	30.980593	15.0	52.0	69.5	91.0	202.0	69.5
max_salary	742.0	128.149596	45.220324	16.0	96.0	124.0	155.0	306.0	124.0
avg_salary	742.0	100.626011	38.855948	13.5	73.5	97.5	122.5	254.0	97.5
same_state	742.0	0.557951	0.496965	0.0	0.0	1.0	1.0	1.0	1.0
age	742.0	46.591644	53.778815	-1.0	11.0	24.0	59.0	276.0	24.0
python_yn	742.0	0.528302	0.499535	0.0	0.0	1.0	1.0	1.0	1.0
R_yn	742.0	0.002695	0.051882	0.0	0.0	0.0	0.0	1.0	0.0
spark	742.0	0.225067	0.417908	0.0	0.0	0.0	0.0	1.0	0.0
aws	742.0	0.237197	0.425651	0.0	0.0	0.0	0.0	1.0	0.0
excel	742.0	0.522911	0.499812	0.0	0.0	1.0	1.0	1.0	1.0
desc_len	742.0	3869.545822	1521.495868	407.0	2801.0	3731.0	4740.0	10051.0	3731.0
num_comp	742.0	1.053908	1.384239	0.0	0.0	0.0	3.0	4.0	0.0

Por ello reemplazamos dichos valores con la mediana de dicho atributo

```
[47] for i in range(0, len(data)):
       if data['Rating'][i]==-1:
         data['Rating'][i]=data['Rating'].median()
       if data['Founded'][i]==-1:
         data['Founded'][i]=data['Founded'].median()
       if data['age'][i]<18 or data['age'][i]>60:
         data['age'][i]=data['age'].median()
```

```
stats_df = data.describe()
# Obtener y añadir la mediana
median = pd.DataFrame(data.median())
median = median.transpose()
median.rename(index = {0:'median'},inplace = True)
stats_df = stats_df.append(median).transpose() # Hay q igualarlo porq append() devuelve nuevo df
stats_df
```

```
<ipython-input-48-d1f1a0e2c2cc>:3: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with
  median = pd.DataFrame(data.median())
```

	count	mean	std	min	25%	50%	75%	max	median
Rating	742.0	3.688544	0.566106	1.9	3.3	3.7	4.0	5.0	3.7
Founded	742.0	1971.184636	52.428471	1744.0	1961.0	1988.0	2007.0	2019.0	1988.0
hourly	742.0	0.032345	0.177034	0.0	0.0	0.0	0.0	1.0	0.0
employer_provided	742.0	0.022911	0.149721	0.0	0.0	0.0	0.0	1.0	0.0
min_salary	742.0	74.719677	30.980593	15.0	52.0	69.5	91.0	202.0	69.5
max_salary	742.0	128.149596	45.220324	16.0	96.0	124.0	155.0	306.0	124.0
avg_salary	742.0	100.626011	38.855948	13.5	73.5	97.5	122.5	254.0	97.5
same_state	742.0	0.557951	0.496965	0.0	0.0	1.0	1.0	1.0	1.0
age	742.0	28.035040	9.314355	18.0	24.0	24.0	24.0	59.0	24.0
python_yn	742.0	0.528302	0.499535	0.0	0.0	1.0	1.0	1.0	1.0
R_yn	742.0	0.002695	0.051882	0.0	0.0	0.0	0.0	1.0	0.0
spark	742.0	0.225067	0.417908	0.0	0.0	0.0	0.0	1.0	0.0
aws	742.0	0.237197	0.425651	0.0	0.0	0.0	0.0	1.0	0.0
excel	742.0	0.522911	0.499812	0.0	0.0	1.0	1.0	1.0	1.0
desc_len	742.0	3869.545822	1521.495868	407.0	2801.0	3731.0	4740.0	10051.0	3731.0
num_comp	742.0	1.053908	1.384239	0.0	0.0	0.0	3.0	4.0	0.0

Hay columnas con medias muy bajas, es decir, casi todo son ceros, por lo que no aportan gran información. Por ello se van a eliminar las columnas de hourly, employer_provided y R_yn.

Además, la columna de desc_len contiene la longitud de la descripción del puesto de trabajo, que para nuestro estudio no es significativa, por lo que eliminamos esta columna.

Realizamos la correlación de las diferentes columnas.

Matriz de correlación de datos numéricos

```
[50] var_num = data.select_dtypes(include = ['int64','float64']).reset_index(drop = True)
     corr = var_num.corr()
     corr.style.background_gradient (cmap = 'coolwarm')
```

	Rating	Founded	max_salary	same_state	age	python_yn	spark	aws	excel	num_comp
Rating	1.000000	0.093924	0.108163	0.004521	-0.098408	0.164118	0.155802	0.149625	-0.023634	-0.055353
Founded	0.093924	1.000000	-0.033656	0.194727	-0.005723	0.116023	0.114081	0.031658	-0.044889	-0.098228
max_salary	0.108163	-0.033656	1.000000	-0.032784	-0.131185	0.301481	0.171317	0.170911	-0.067175	0.086195
same_state	0.004521	0.194727	-0.032784	1.000000	0.074779	0.006975	-0.053139	-0.065070	0.106026	-0.102635
age	-0.098408	-0.005723	-0.131185	0.074779	1.000000	-0.114200	-0.133080	-0.142680	-0.011188	-0.024953
python_yn	0.164118	0.116023	0.301481	0.006975	-0.114200	1.000000	0.347619	0.203221	-0.021519	0.091470
spark	0.155802	0.114081	0.171317	-0.053139	-0.133080	0.347619	1.000000	0.298822	-0.047334	0.069980
aws	0.149625	0.031658	0.170911	-0.065070	-0.142680	0.203221	0.298822	1.000000	-0.019235	-0.014860
excel	-0.023634	-0.044889	-0.067175	0.106026	-0.011188	-0.021519	-0.047334	-0.019235	1.000000	-0.031046
num_comp	-0.055353	-0.098228	0.086195	-0.102635	-0.024953	0.091470	0.069980	-0.014860	-0.031046	1.000000

No vemos correlaciones importantes entre los campos numéricos.

4. Análisis de datos

- Selección de los grupos de datos que se quieren analizar/comparar (p. ej., si se van a comparar grupos de datos, ¿cuáles son estos grupos y qué tipo de análisis se van a aplicar?)

  En este caso vamos a comparar los sueldos de los diferentes job_title. Por ello vamos a coger los 2 puestos de trabajo más ofertados y vamos a hacer contrastes de los sueldos máximos.

```
[52] data.groupby(['Job Title'])['Job Title'].count().reset_index(name='count').sort_values(['count'], ascending=False).head(5)
```

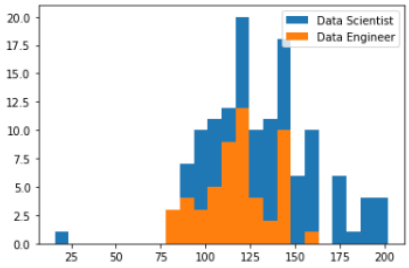
	Job Title	count
69	Data Scientist	131
51	Data Engineer	53
195	Senior Data Scientist	34
40	Data Analyst	15
193	Senior Data Engineer	14

Vamos a optar por el los puestos de Data Scientist y Data Engineer.

```
[53] df_DS = data[data['Job Title']=='Data Scientist']
     df_DE = data[data['Job Title']=='Data Engineer']
```

Comprobamos la normalidad y homogeneidad de la varianza.

```
[54] bins = np.linspace(min(df_DS['max_salary']), max(df_DS['max_salary']), 25)
     plt.hist(df_DS['max_salary'], bins, label='Data Scientist')
     plt.hist(df_DE['max_salary'], bins, label='Data Engineer')
     plt.legend(loc='upper right')
     plt.show()
```



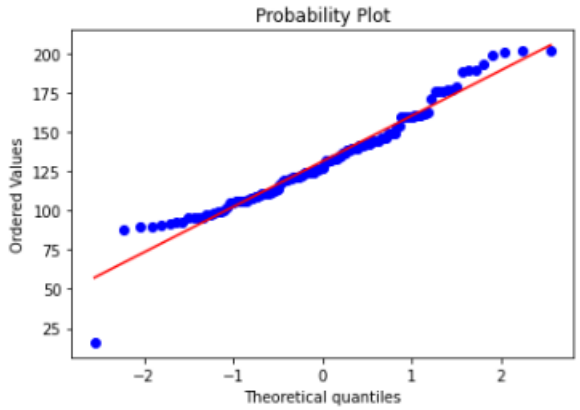
Los grupos para analizar serán los puestos de trabajo de Data Scientist y Data Engineer.

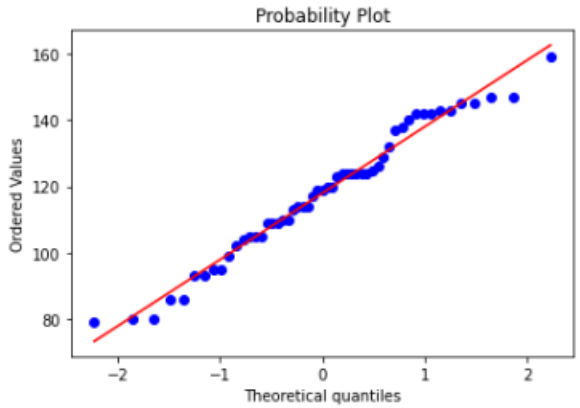
- Comprobación de la normalidad y homogeneidad de la varianza.
  Comprobamos la normalidad utilizando el test de Shapiro-Wilk en los sueldos máximos.

```
from scipy import stats
import numpy as np
import pylab
print(stats.shapiro(df_DS['max_salary']))
print(stats.shapiro(df_DE['max_salary']))

stats.probplot(df_DS['max_salary'],dist="norm", plot=pylab)
pylab.show()
stats.probplot(df_DE['max_salary'],dist="norm", plot=pylab)
pylab.show()
```

```
ShapiroResult(statistic=0.9604859948158264, pvalue=0.0007525604451075196)
ShapiroResult(statistic=0.9733689427375793, pvalue=0.2809053361415863)
```





Dado que la prueba de Shapiro-Wilk se considera más robusta, una posición más conservadora concluiría que los datos para el puesto de Data Scientist no siguen una distribución normal. No obstante, como el conjunto de datos supera los 30 registros, por el teorema central del límite, se podría considerar que los datos siguen una distribución normal. Los datos para el puesto de Data Engineer sí siguen una distribución normal.

Comprobamos la homocedasticidad gracias al test de Levene y Fligner.

```
[56] print(stats.levene(df_DS['max_salary'],df_DE['max_salary']))
     print(stats.fligner(df_DS['max_salary'],df_DE['max_salary']))
```

```
LeveneResult(statistic=5.793675375333901, pvalue=0.017083602924103082)
FlignerResult(statistic=4.876003494161175, pvalue=0.027232602411892067)
```

Dado que ambas pruebas resultan en un p-valor inferior al nivel de significancia (< 0,05), se rechaza la hipótesis nula de homocedasticidad y se concluye que la variable

max_salary presenta varianzas estadísticamente diferentes para los diferentes grupos de Job Title.

- Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

  Nuestra hipótesis es que el salario máximo de los Data Scientist es superior a los Data Engineers, es decir, esta es nuestra hipótesis alternativa. Por ello, vamos a usar el T-test con varianzas diferentes para comprobarlo.

```
[57] stats.ttest_ind(df_DS['max_salary'],df_DE['max_salary'],equal_var=False,alternative='greater')

     Ttest_indResult(statistic=3.588214204500318, pvalue=0.0002292133435459003 6)
```

  Como el p-value es menor a 0.05, aceptamos la hipótesis alternativa y concluimos que tenemos suficientes evidencias para afirmar que el salario máximo de los Data Scientist es superior a los Data Engineers.

  Realizamos las correlaciones de los diferentes grupos:

```
[58] corr = df_DS.corr()
     corr.style.background_gradient (cmap = 'coolwarm')
```

	Rating	Founded	max_salary	same_state	age	python_yn	spark	aws	excel	num_comp
Rating	1.000000	0.144958	-0.068531	0.104538	0.041799	-0.041272	0.090956	-0.009795	0.163604	0.018573
Founded	0.144958	1.000000	0.075834	0.223389	-0.061521	0.027213	0.178904	0.214266	0.019529	-0.062326
max_salary	-0.068531	0.075834	1.000000	-0.051236	-0.249044	-0.032423	0.229334	0.068739	-0.229161	0.073967
same_state	0.104538	0.223389	-0.051236	1.000000	0.051585	0.030938	0.015830	-0.110387	0.063306	0.037862
age	0.041799	-0.061521	-0.249044	0.051585	1.000000	-0.170154	-0.163832	-0.045711	-0.017858	-0.014189
python_yn	-0.041272	0.027213	-0.032423	0.030938	-0.170154	1.000000	0.233021	0.065306	-0.085102	0.226120
spark	0.090956	0.178904	0.229334	0.015830	-0.163832	0.233021	1.000000	0.310418	-0.124807	0.032811
aws	-0.009795	0.214266	0.068739	-0.110387	-0.045711	0.065306	0.310418	1.000000	-0.119228	0.026231
excel	0.163604	0.019529	-0.229161	0.063306	-0.017858	-0.085102	-0.124807	-0.119228	1.000000	-0.088714
num_comp	0.018573	-0.062326	0.073967	0.037862	-0.014189	0.226120	0.032811	0.026231	-0.088714	1.000000

```
[59] var_num = df_DE.select_dtypes(include = ['int64','float64']).reset_index(drop = True)
     corr = var_num.corr()
     corr.style.background_gradient (cmap = 'coolwarm')
```

	Rating	Founded	max_salary	same_state	age	python_yn	spark	aws	excel	num_comp
Rating	1.000000	0.066094	0.080763	0.013163	0.056465	-0.129415	0.217122	-0.097082	-0.292225	-0.229290
Founded	0.066094	1.000000	-0.039236	-0.320672	-0.121798	-0.000649	0.171020	0.178052	-0.095882	0.206773
max_salary	0.080763	-0.039236	1.000000	-0.085896	0.033596	-0.048740	-0.059266	0.097582	-0.138060	-0.272542
same_state	0.013163	-0.320672	-0.085896	1.000000	0.207758	0.182521	-0.173857	0.095578	0.207977	-0.042400
age	0.056465	-0.121798	0.033596	0.207758	1.000000	-0.270219	-0.135596	-0.293724	-0.195537	-0.088721
python_yn	-0.129415	-0.000649	-0.048740	0.182521	-0.270219	1.000000	0.425226	0.223220	0.159908	0.134263
spark	0.217122	0.171020	-0.059266	-0.173857	-0.135596	0.425226	1.000000	-0.011511	0.021553	0.087620
aws	-0.097082	0.178052	0.097582	0.095578	-0.293724	0.223220	-0.011511	1.000000	0.131241	-0.037282
excel	-0.292225	-0.095882	-0.138060	0.207977	-0.195537	0.159908	0.021553	0.131241	1.000000	0.289042
num_comp	-0.229290	0.206773	-0.272542	-0.042400	-0.088721	0.134263	0.087620	-0.037282	0.289042	1.000000

  Vemos en los 2 casos que no aparecen correlaciones fuertes entre max_salary y el resto de los atributos. Lo que nos parece interesante es que en el grupo de Data Engineer

aparece cierta relación entre la demanda de Python y spark, es decir, es frecuente que pidan estos 2 skills para este tipo de trabajo.

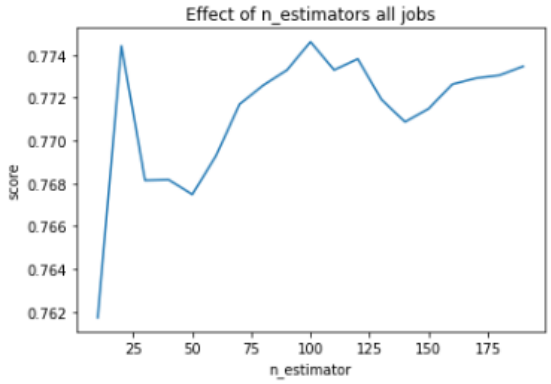
Finalmente realizamos un modelo de regresión para predecir el salario máximo usando un algoritmo RandomForest ya que usa atributos categóricos como numéricos para cada uno de los job_title.

```
[62] X = data.drop(['max_salary'],axis=1)
     y = data['max_salary']

     X_encoded = pd.get_dummies(X, drop_first=True)
     X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)

     regressor = RandomForestRegressor(random_state=42)
     estimators = np.arange(10, 200, 10)
     scores = []
     for n in estimators:
         regressor.set_params(n_estimators=n)
         regressor.fit(X_train, y_train)
         scores.append(regressor.score(X_test, y_test))
     plt.title("Effect of n_estimators all jobs")
     plt.xlabel("n_estimator")
     plt.ylabel("score")
     plt.plot(estimators, scores)
```

```
[<matplotlib.lines.Line2D at 0x7f40aabd30a0>]
```



Podemos ver que el modelo de regresión propuesto alcanza un accuracy elevado con un valor bajo de estimadores. Si aumentamos dicho parámetro obtenemos una mejora del accuracy que no es importante.

5. Representación de los resultados

Este apartado se puede responder a lo largo de la práctica, sin necesidad de concentrar todas las representaciones en este punto de la práctica.

6. Resolución del problema

A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Hemos realizado varios procesos de limpieza, selección y filtrado de características con el objetivo de analizar en qué puesto se obtiene un mayor salario, si en Data Scientist o en Data Engineer.

Con el contraste de hipótesis se obtiene que el rol de Data Scientist está mejor remunerado que el de Data Engineer. Además, hemos creado un modelo de regresión que predice el salario máximo en función de las características del puesto de trabajo.

7. Vídeo

Link de acceso al vídeo:

https://drive.google.com/file/d/1DsrJ-43GMi4jrMbAWpRRLmqllYMY5-1n/view?usp=sharing

Contribuciones	Firma
Investigación previa	**AAS, CNA**
Redacción de las respuestas	**AAS, CNA**
Desarrollo del código	**AAS, CNA**
Participación en el vídeo	**AAS, CNA**