TRABAJO PRACTICO N°1 – PANDAS Y VISUALIZACION DE DATOS

ORGANIZACIÓN DE DATOS 75.06

Alumno: Gonzalez, Agustin Nicolas

Padrón: 106086

Corrector: Damián.

Link al colab:
https://colab.research.google.com/drive/1ZUaoC9A7ECenvGvDdD4ikM3BVjyfxWzx?usp=sharing

Primera parte:

33. La primera discusión creada

```
#33
acciones_creadas = acciones[acciones["action"]=="create"]
discusiones_creadas = acciones_creadas[acciones_creadas["comment"].str.contains("Discusion", na=False)]
primera_discusion = discusiones_creadas.iloc[0]
"La primera discusion fue creada por " + primera_discusion["contributor_username"] + " con fecha " + primera_discusion["timestamp"]
```

```
'La primera discusion fue creada por IngenieroLoco con fecha 2018-07-07T13:05:43Z'
```

7. La antigüedad promedio de la última edición de los artículos cuyo título contenga tu apellido (si no hay, tu nombre y si tampoco hay usa Cafferata)

```
[ ] #7
    gonzalez = contenidos[contenidos.title.str.contains("Gonzalez", na=False)]
    gonzalez["revision_timestamp"] = pd.to_datetime(gonzalez["revision_timestamp"],format="%Y-%m-%d")
    promedio = gonzalez["revision_timestamp"].mean()
    antiguedad = pd.to_datetime("2020-04-11 19:00:00.0000+0000") - pd.to_datetime(promedio)
    "La antiguedad promedio es de " + str(antiguedad.days) + " dias."
```

```
'La antiguedad promedio es de 3185 dias.'
```

9. Cuáles son los contenidos de wikipedia cuyo título empieza o termina con un emoji

```
[ ] #9
    import emoji

    contenidos_filtrados = contenidos.fillna(" ")
    empieza = contenidos_filtrados[contenidos_filtrados["title"].str.startswith(tuple(emoji.UNICODE_EMOJI_SPANISH))]
    empieza_y_termina = empieza[empieza["title"].str.endswith(tuple(emoji.UNICODE_EMOJI_SPANISH))]
    empieza_y_termina["title"]
```

```
102653      ®
339096      ©
339097      ™
481342      ✈
481348      🅿
            ..
3722169     SA
3841251     🤓
3966243     🅽
4101129     ✅
4101131     ✓
Name: title, Length: 87, dtype: object
```

Aclaracion: Se que estoy perdiendo efectividad al chequar primero con startswith y después con endswith, que convendría hacerlo todo en un paso, pero cuando lo intentaba me saltaba error

11. Para todos los comentarios de revisión de contenido que tengan más de 20 ocurrencias realice una matriz cuyas columnas sean esos comentarios y de índice los usuarios/ips con valores: True si ese usuario realizó ese comentario, sino False

```
#11
contenidos["usuarios"] = contenidos["revisor_username"]
contenidos["usuarios"].fillna(contenidos["revisor_ip"],inplace=True)
ocurrencias = contenidos["revisor_comment"].value_counts().rename_axis('revisor_comment').reset_index(name='counts')
ocurrencias = ocurrencias.loc[ocurrencias["counts"]>20,"revisor_comment"]
union = pd.merge(ocurrencias,contenidos)
resultado = union.groupby(["revisor_comment", "usuarios"]).count().reset_index()
matriz = resultado.pivot(columns="revisor_comment", index="usuarios", values="id")
matriz.notnull()
```

Aclaración, se corta la visualización de la respuesta ya que la matriz es demasiado grande para ser capturada por completo. Sobre este ejercicio también se asumió que todos los usuarios a ser incluidos en la matriz son todos los que realizaron al menos alguno de esos comentarios, por eso la amplia longitud de la matriz.

revisor_comment	#QuédateEnTuCasaYEditaWikipedia. Nuevo, localidad	#REDIRECCIÓN	#REDIRECT	#REDIRECT [[Anexo:Glosario de términos de sumo]]	#REDIRECT [[Echinopsis]]	#REDIRECT [[Wikipedia:Consultas de borrado/Lista de universidades por país]]	#REDIRECT [[]]	#WPWP	#WPWP #WPWPARK	#WPWP #WPWPSR	(Bot) Correcciones en plantillas	(Bot) Correcciones ortográficas	(Bot) Correcciones ortográficas; cambios superficiales
usuarios													
"BF CLUB"	False	False	False	False	False	False	False	False	False	False	False	False	False
"CLUB BF"	False	False	False	False	False	False	False	False	False	False	False	False	False
&beer&love	False	False	False	False	False	False	False	False	False	False	False	False	False
(:Julien:)	False	False	False	False	False	False	False	False	False	False	False	False	False
(KGC2) KosmoGelo II	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
熊	False	False	False	False	False	False	False	False	False	False	False	False	False
猫に小判	False	False	False	False	False	False	False	False	False	False	False	False	False
諾恩	False	False	False	False	False	False	False	False	False	False	False	False	False
阿道	False	False	False	False	False	False	False	False	False	False	False	False	False
隼鷹	False	False	False	False	False	False	False	False	False	False	False	False	False

29937 rows × 6180 columns

17. Utilice los textos del contenido para realizar consultas por texto utilizando las técnicas vistas en la clase de NLP (BOW o TF-IDF) de modo que la query "retablo iglesia" devuelva alguna página acerca del retablo de alguna iglesia

Aclaración sobre este ejercicio, se probó también a hacer lo recomendado en una de las respuestas a mi consulta de buscar primero una query retablo y otra con la query iglesia y luego hacer un merge, pero el resultado fue mas pobre que el obtenido en este caso, también se probo a aumentar y reducir los max_features, pero esta fue la respuesta mas optima que se encontró.

```
[ ] #17
    import nltk
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.neighbors import NearestNeighbors
    textos = textos_de_contenidos["text"].fillna(" ")
    vectorizer = CountVectorizer(lowercase=True, stop_words=nltk.corpus.stopwords.words('spanish'), max_features=20000)
    matriz = vectorizer.fit_transform(textos)
    buscador = NearestNeighbors(algorithm="brute")
    buscador.fit(matriz)
    query = "retablo iglesia"
    vector = vectorizer.transform([query])
    dist, indices = buscador.kneighbors(vector, n_neighbors=10)
    for d, i in zip(dist.flatten(), indices.flatten()):
      print(textos[i])
```

```
#REDIRECCIÓN [[Retablo de Bernabé]]
#REDIRECCION [[Iglesia de la Asunción (Alcocer)]]
|}
#REDIRECCIÓN [[Retablo]]
#REDIRECCION[[Badou Ndiaye]]
#REDIRECCION[[Aerofobia]]

#REDIRECT [[Iglesia de Täby]]
#REDIRECCION [[Kufiyya]]
#REDIRECCION [[Estalagmita]]
```

41. Para los contenidos geolocalizados, según la última versión de cada contenido: ¿Cuál es la latitud y longitud promedio del contenido editado según qué idioma sabe el editor?

```
[ ] #41
    localizacion = geo.drop_duplicates("gt_page_id", keep="first")
    ubicaciones = pd.merge(contenidos, localizacion, left_on="id", right_on="gt_page_id")
    total = pd.merge(lenguages, ubicaciones, left_on="babel_user", right_on="revisor_id")
    total = total.loc[:,["babel_lang", "gt_lat", "gt_lon"]]
    total.groupby("babel_lang").mean()
```

babel_lang	gt_lat	gt_lon
ab	37.067389	-0.701931
af	52.010278	-8.610278
agr	-11.610633	-76.309961
akk	-11.610633	-76.309961
an	39.067374	6.994832
...	...	...
uz	-11.610633	-76.309961
vec	42.934824	10.522120
vi	5.524444	-68.685556
xcw	21.393306	-103.188056
zh	27.525063	75.491088

129 rows × 2 columns

49. Si decimos que un usuario sabe un idioma cuando tiene un nivel de babel mayor o igual a 1, para aquellos que editaron una de las versiones actuales del contenido, ¿Cuál es la tasa de revisiones sin comentario que realizan en función de los idiomas que saben?

```
[ ] #49
    #lenguage = lenguages.loc[lenguages["babel_level"] != "N",:]
    lenguage_sabido = lenguages.loc[lenguages["babel_level"]!="0",:]
    contador_idiomas = lenguage_sabido["babel_user"].value_counts().reset_index(name="cantidad_idiomas")
    contenidos_revisados = pd.merge(contenidos, lenguage_sabido, left_on="revisor_id", right_on="babel_user")
    contenidos_revisados.loc[contenidos_revisados["revisor_comment"].isnull(),:]
    contador_revisados = contenidos_revisados["babel_user"].value_counts().reset_index(name="cantidad_revisiones")
    union = pd.merge(contador_revisados, contador_idiomas)
    union["tasa"] = union["cantidad_revisiones"]/union["cantidad_idiomas"]
    union.rename(columns={"index": "usuario"}, inplace=True)
    union.loc[:,["usuario", "tasa"]]
```

	usuario	tasa
0	58607	35837.0
1	13349	28163.0
2	474907	37975.0
3	2075	22481.0
4	998086	3416.0
...	...	...
4626	597205	1.0
4627	2209373	1.0
4628	1636034	1.0
4629	233719	1.0
4630	3576503	1.0

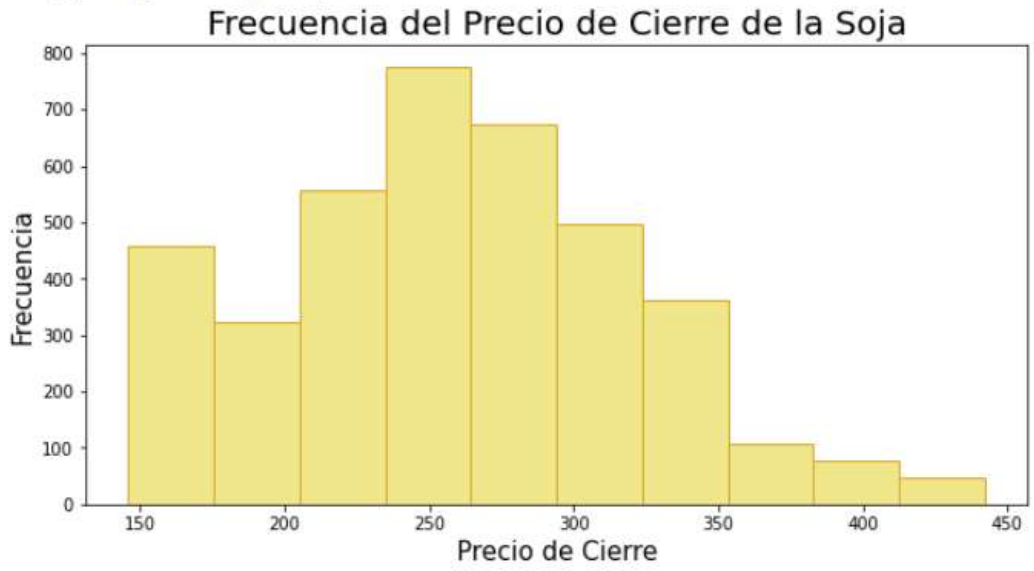
4631 rows × 2 columns

Segunda parte:

Dataset: Proyectando el comportamiento de la soja

Tipo de plot: Histograma

```
Text(0, 0.5, 'Frecuencia')
```



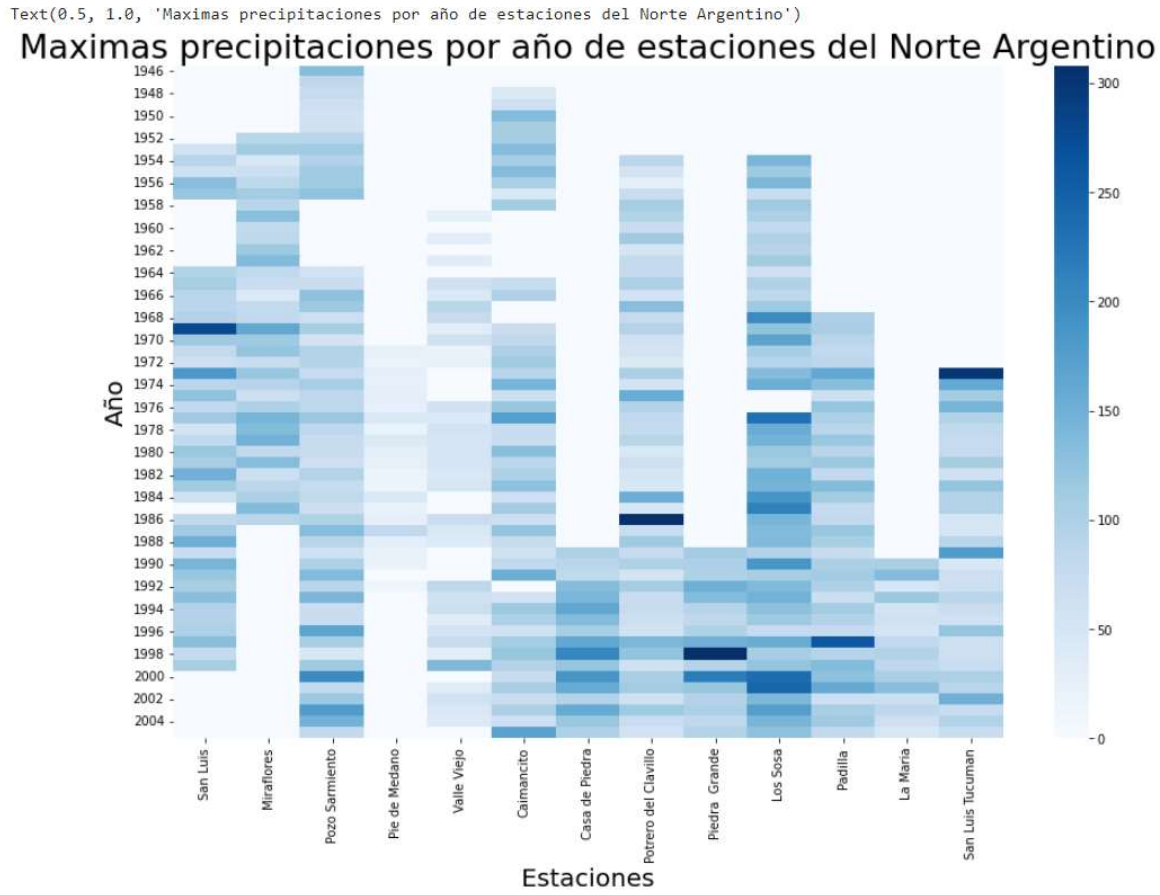
Tipo de plot: Lineal

```
<Figure size 720x360 with 0 Axes>
```

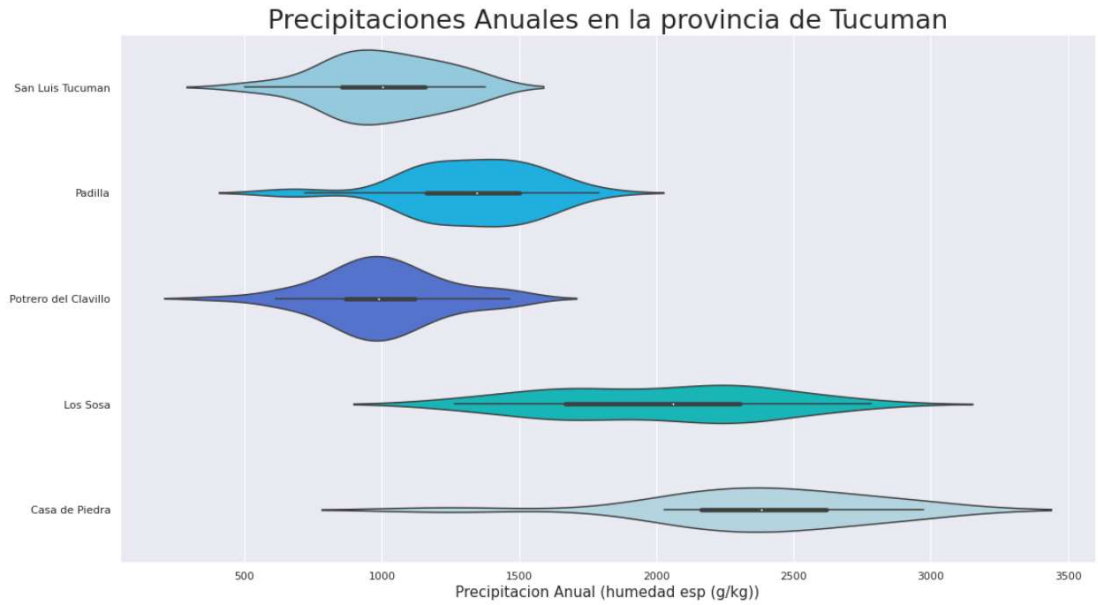

```
<Figure size 720x360 with 0 Axes>
```

Dataset: ¿Llevo paraguas? Pronosticando la lluvia

Tipo de plot: Heatmap

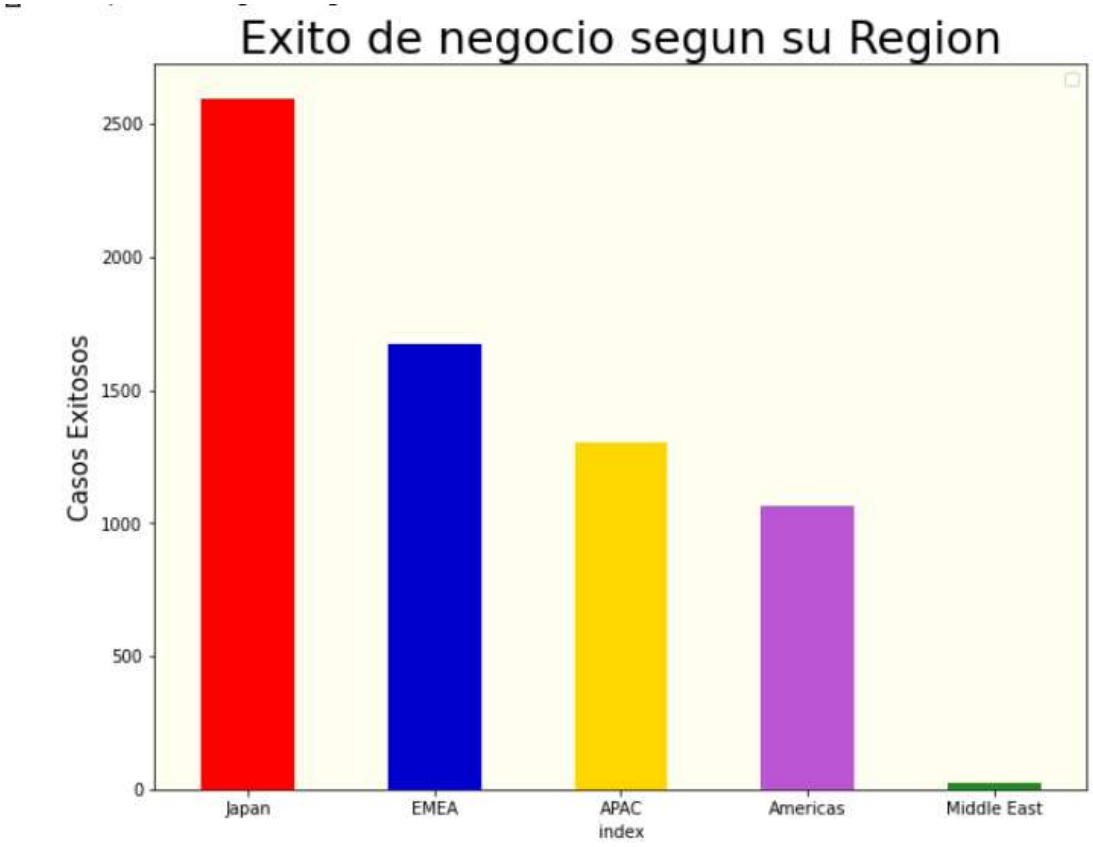


Tipo de plot: Violín

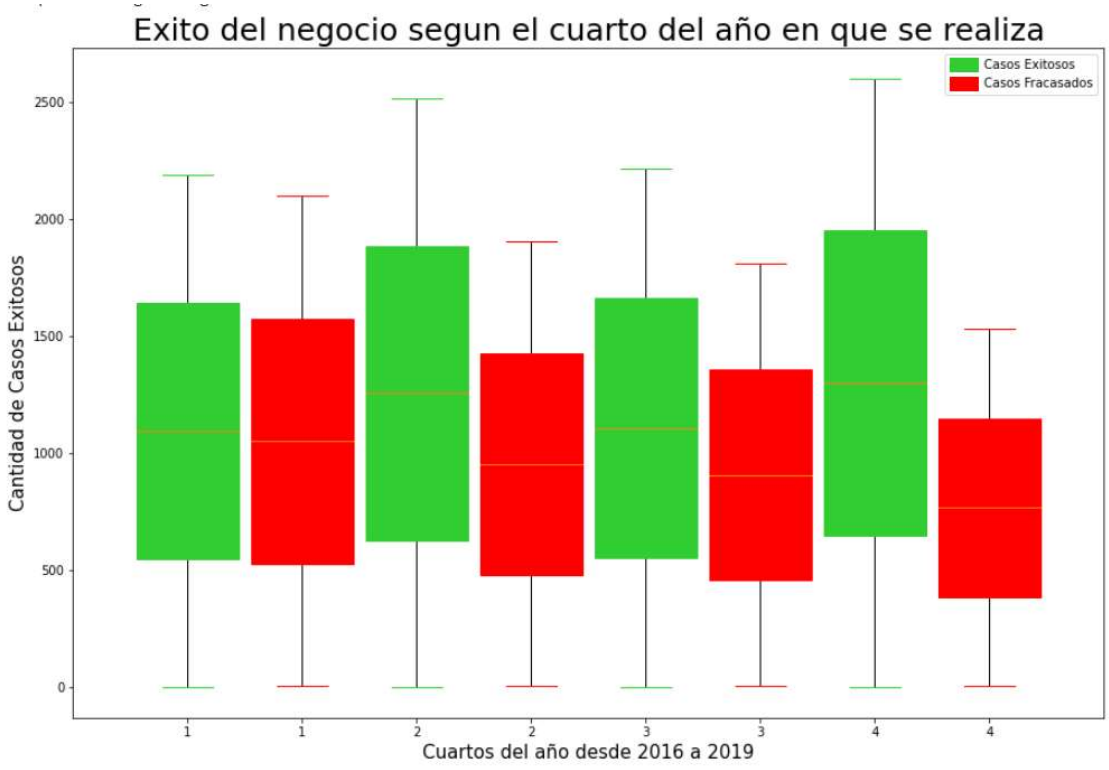


Dataset: Predicción de éxitos en oportunidades laborales.

Tipo de plot: Bar-Plot



Tipo de plot: Boxplot



En el siguiente plot se tomo como referencia uno de los dataset provistos por la pagina de datasets datos del gobierno. Se trata de plasmar cuales son las cárceles mas pobladas de la provincia de Buenos Aires, se filtro Capital y el resto de las provincias ya que no se podría discernir tantas diferencias en contextos tan diferentes. En esta corrección me di cuenta del fallo de solo haber tenido en cuenta a los presos imputados (por eso en esta visu se puede ver que la cantidad media de presos es mayor), pero esta vez se tomo en cuenta a los imputados y procesados, y agregándose la capacidad de estas mismas unidades penitenciarias, ahora nos damos cuenta de cuales tienen mas sobrepoblacion, y cuales tienen mas espacios libres, proponiendo quizá, una posible reorganización de estas unidades penitenciarias.