Analysis of University Admissions Data

Entonces, analicemos los datos y predigamos la probabilidad de admisión del estudiante en una universidad en particular en función de varios parámetros.

Las diferentes entidades o parámetros en el conjunto de datos son:

- Serial No : Identificador del aspirante.
- GRE Scrore : Puntaje de la prueba GRE, que es una prueba importante para la admisión en el proceso de solicitud de la escuela de posgrado o la escuela de negocios a nivel mundial.
- TOEFL Score : Prueba de puntuación del examen de inglés como lengua extranjera.
- Universiting Rating : Calificación de la Universidad sobre 5.
- SOP : Relacionado con la Declaración de Propósito (SOP) para aplicar a un curso o universidad en particular.
- LOR : Algún puntaje relacionado con LOR, es decir, una carta de recomendación.
- CGPA : Es una medida de desempeño anterior del aspirante.
- Research : Binary values of either 1 or 0.
- Chance of Admit : Probabilidad de que el estudiante entre.

Importarmos las librerias

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import Latex, Math, display
from sklearn.linear_model import LinearRegression
from IPython.display import Image
import statsmodels.formula.api as smf
```

Python

Cargamos los Datos

```python
data = pd.read_csv(r"C:\Users\guill\Desktop\p\MC\Temas\tareas\adm_data.csv", index_col= 0)
data.head()
```

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
Serial No.								
1	337	118	4	4.5	4.5	9.65	1	0.92
2	324	107	4	4.0	4.5	8.87	1	0.76

Serial No.								
1	337	118	4	4.5	4.5	9.65	1	0.92
2	324	107	4	4.0	4.5	8.87	1	0.76
3	316	104	3	3.0	3.5	8.00	1	0.72
4	322	110	3	3.5	2.5	8.67	1	0.80
5	314	103	2	2.0	3.0	8.21	0	0.65

Se crea una lista donde se guardara el nombre de las columnas

```
x = data.columns.values.tolist()
y = "Chance_of_Admit"
```

[3] Python

Se revisa que los datos no haya un N/A

```
null = []
for i in x:
    null.append(pd.isnull(data[i]).values.ravel().sum())
null
```

Plots

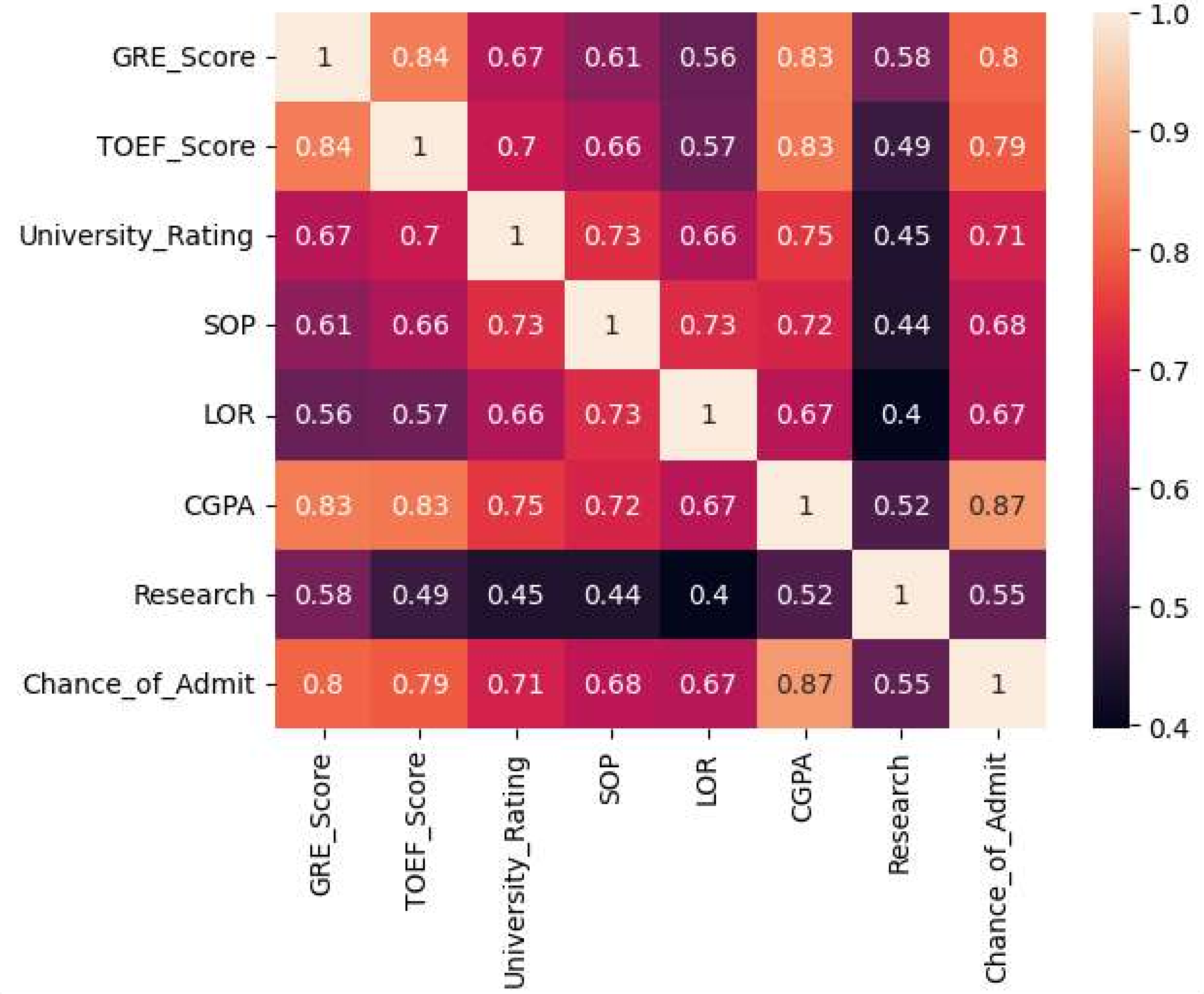
Creación de mapa de calor

```
lista = []
aux = []
for i in x:
    for j in x:
        aux.append(coeficiente_pearson(data,i,j))
    lista.append(aux)
    aux = []
lista
matriz_pearson = np.array(lista)
matriz_pearson
print("Matriz de coeficientes lineales para cada variable del dataset")
sns.heatmap(matriz_pearson, linecolor="white",annot = True , xticklabels=x, yticklabels=x)
```

[7]

Python

```
Matriz de coeficientes lineales para cada variable del dataset
```


GRE_Score
TOEF_Score
University_Rating
SOP
LOR
CGPA
Research
Chance_of_Admit
1 0.84 0.67 0.61 0.56 0.83 0.58 0.8
0.84 1 0.7 0.66 0.57 0.83 0.49 0.79
0.67 0.7 1 0.73 0.66 0.75 0.45 0.71
0.61 0.66 0.73 1 0.73 0.72 0.44 0.68
0.56 0.57 0.66 0.73 1 0.67 0.4 0.67
0.83 0.83 0.75 0.72 0.67 1 0.52 0.87
0.58 0.49 0.45 0.44 0.4 0.52 1 0.55
0.8 0.79 0.71 0.68 0.67 0.87 0.55 1
1.0
0.9
0.8
0.7
0.6
0.5
0.4

```python
resultado = zip(x,matriz_pearson[7])
print('Esta son los coeficientes de linealidad (Coeficiente de Pearson) de todas las variables respecto a la variable a predecir "Chance of Admit"')
print(list(resultado))
print("Recordando que la prueba de coeficciente de correlación de Pearson nos dice que tanta dependencia lineal hay en un par de variables")
display(Math (r'r_xy = \frac{\sum_{i = 1}^{n} (x_i - \bar{x})(y_i - \bar{y}))}{\sqrt[2]{\sum_{i = 1}^{n} (x_i - \bar{x})^2} \sqrt[2]{\sum_{i = 1}^{n} (y_i - \b
display(Math(r'Donde: \\ \ -1 < r < 1'))
display(Math(r'\begin{cases} r > .6 \Rightarrow Existe \ una \ correlación \ positiva \\ -.6 < r < .6 \Rightarrow \ La \ relación \ es \ casualidad \\ -.6 < r
```

[8] Python

```
Esta son los coeficientes de linealidad (Coeficiente de Pearson) de todas las variables respecto a la variable a predecir "Chance of Admit"
[('GRE_Score', 0.8026104595903508), ('TOEF_Score', 0.7915939869351032), ('University_Rating', 0.7112502503917211), ('SOP', 0.6757318583886718), ('LOR', 0.6698887920106948), ('CGPA', 0.8732890993552993), ('Research', 0.5532021370190391), ('Chance_of_Admit', 1.0)]
Recordando que la prueba de coeficciente de correlación de Pearson nos dice que tanta dependencia lineal hay en un par de variables
```

$$r_xy = \frac{\sum_{i = 1}^{n} (x_i - \bar{x})(y_i - \bar{y}))}{\sqrt[2]{\sum_{i = 1}^{n} (x_i - \bar{x})^2} \sqrt[2]{\sum_{i = 1}^{n} (y_i - \bar{y})^2}}$$

$$Donde: \\ \ -1 < r < 1$$

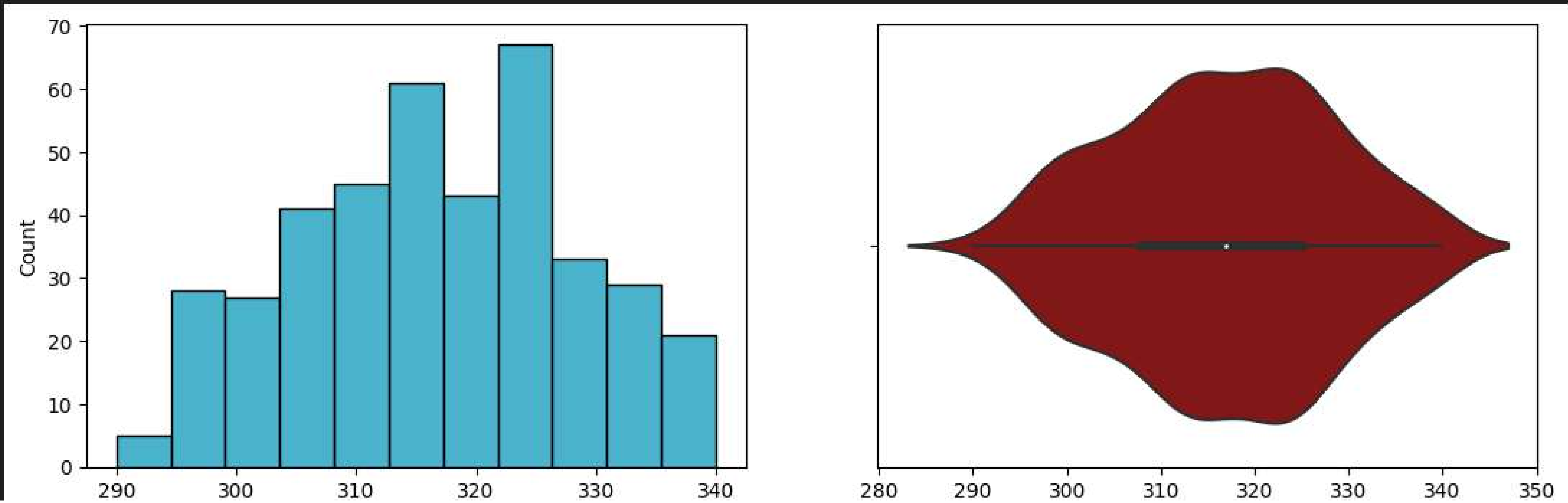
$$\begin{cases} r > .6 \Rightarrow Existe \ una \ correlación \ positiva \\ -.6 < r < .6 \Rightarrow \ La \ relación \ es \ casualidad \\ -.6 < r \Rightarrow Existe \ una \ correlación \ negativa \end{cases}$$

Plots GRE Score

```python
plt.figure(figsize=(13,4))
plt.subplot(1,2,1)
sns.histplot(x="GRE_Score", data=data,color='#0D98BA')
plt.subplot(1,2,2)
sns.violinplot(x = "GRE_Score", data = data, color = '#990000', saturation = .7)
```

[9] Python

```
<AxesSubplot:xlabel='GRE_Score'>
```

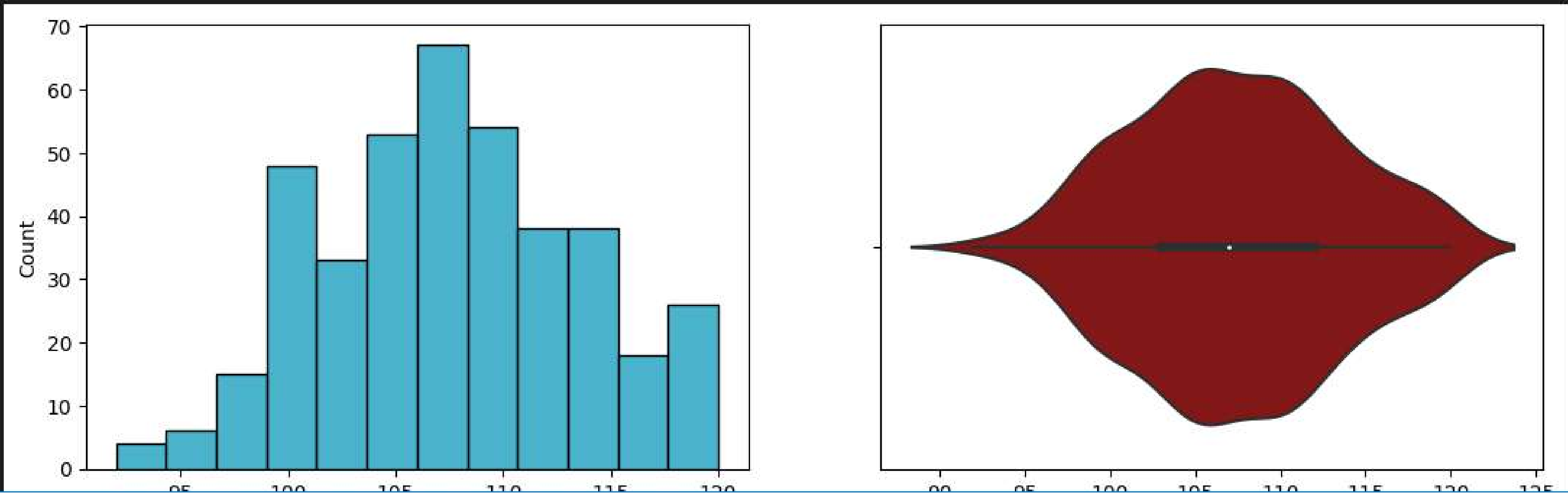


Plots Toefl score

```python
plt.figure(figsize=(13,4))
plt.subplot(1,2,1)
sns.histplot(x="TOEF_Score", data=data,color='#0D98BA')
plt.subplot(1,2,2)
sns.violinplot(x = "TOEF_Score", data = data, color = '#990000', saturation = .7)
```

Python

<AxesSubplot:xlabel='TOEF_Score'>

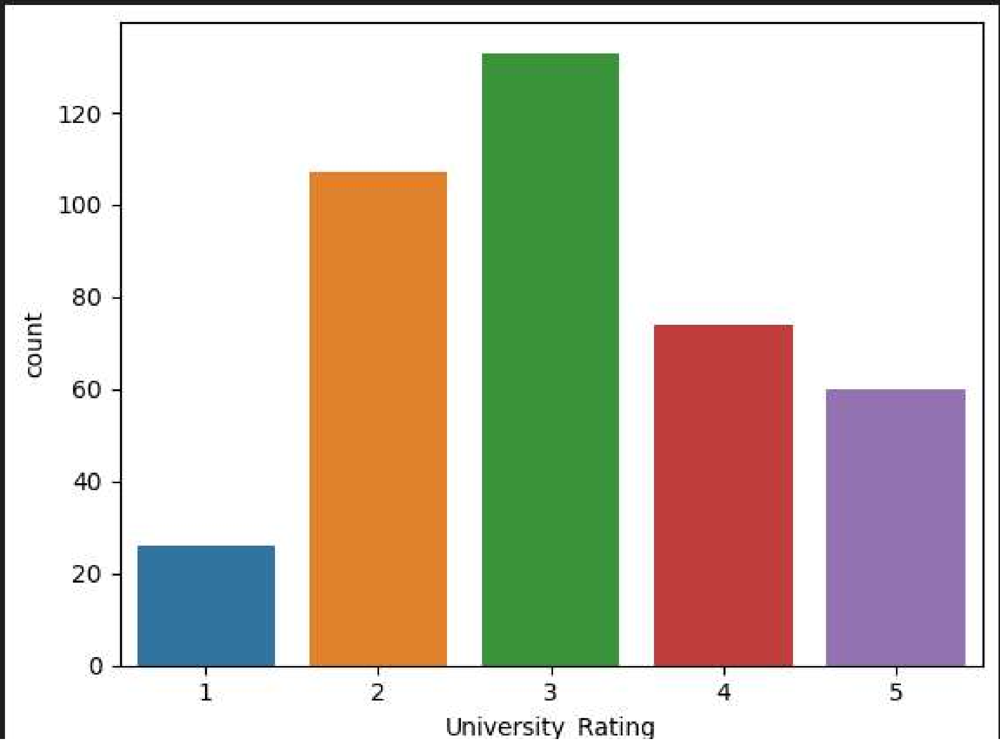


Plots de University Rating

```python
sns.countplot(x=data["University_Rating"])
```

Python

<AxesSubplot:xlabel='University_Rating', ylabel='count'>



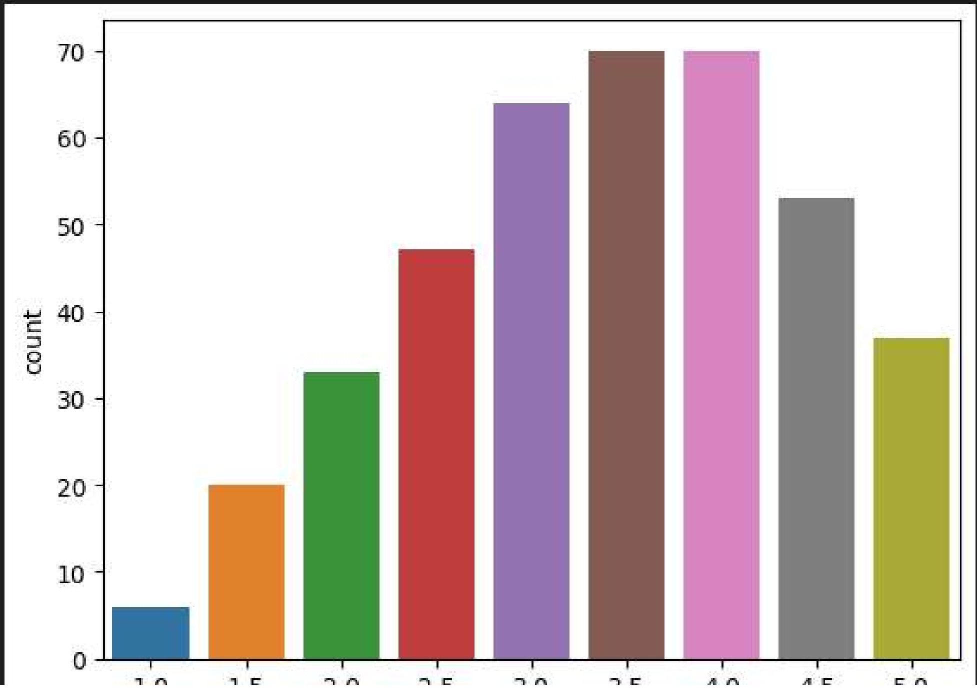
Plots SOP

```
sns.countplot(x=data["SOP"])
```

[12]

Python

<AxesSubplot:xlabel='SOP', ylabel='count'>



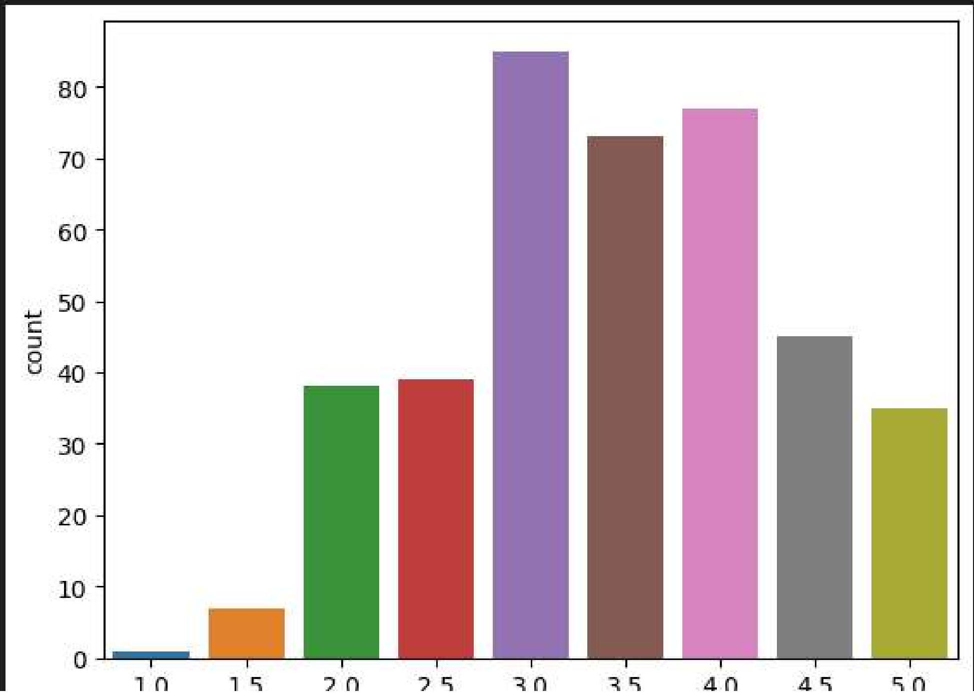
Plots LOR

```
sns.countplot(x=data["LOR"])
```

[13]

Python

<AxesSubplot:xlabel='LOR', ylabel='count'>

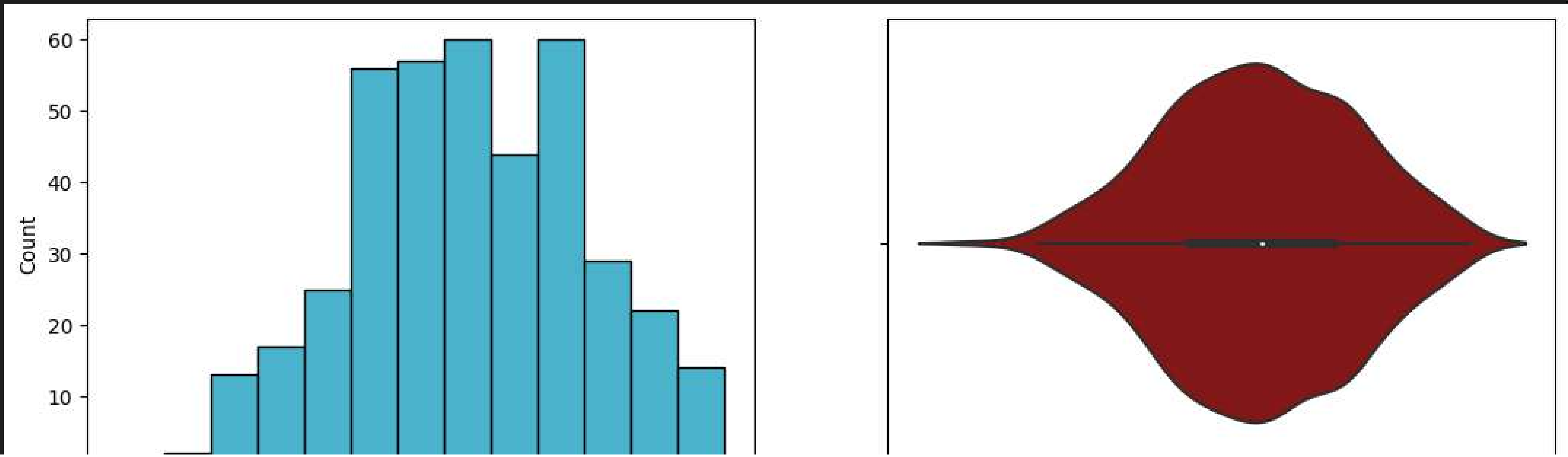


Plots CGPA

```python
plt.figure(figsize=(13,4))
plt.subplot(1,2,1)
sns.histplot(x="CGPA", data=data,color='#0D98BA')
plt.subplot(1,2,2)
sns.violinplot(x = "CGPA", data = data, color = '#990000', saturation = .7)
```

Python

```
<AxesSubplot:xlabel='CGPA'>
```



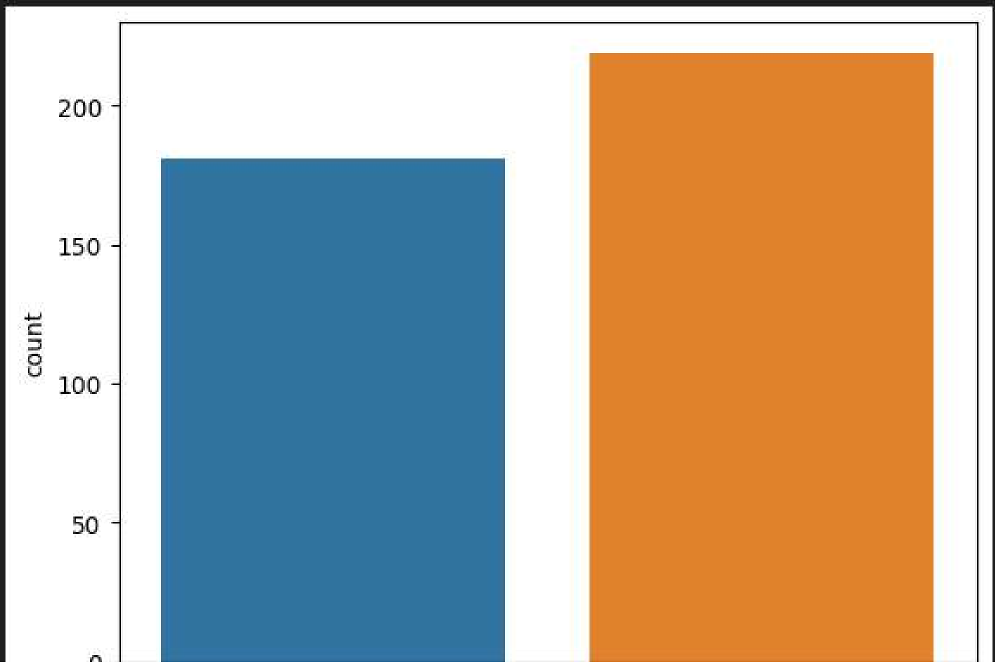
Plot Research

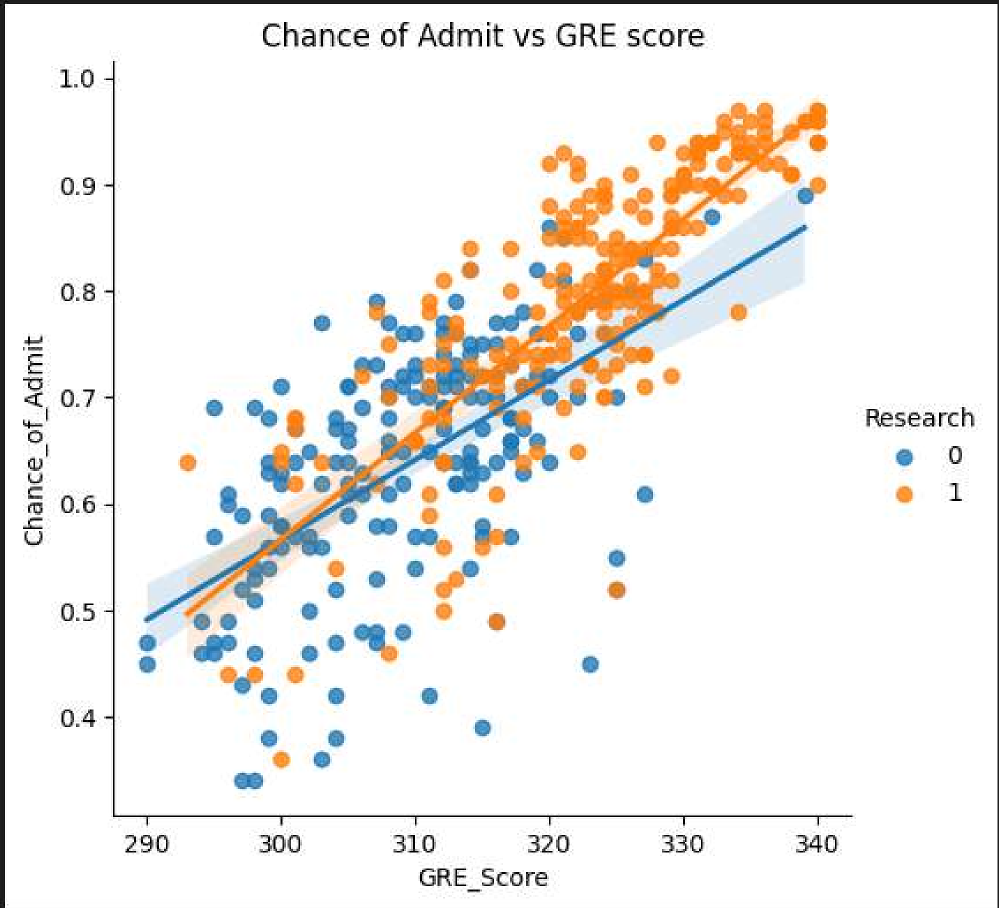
[15]

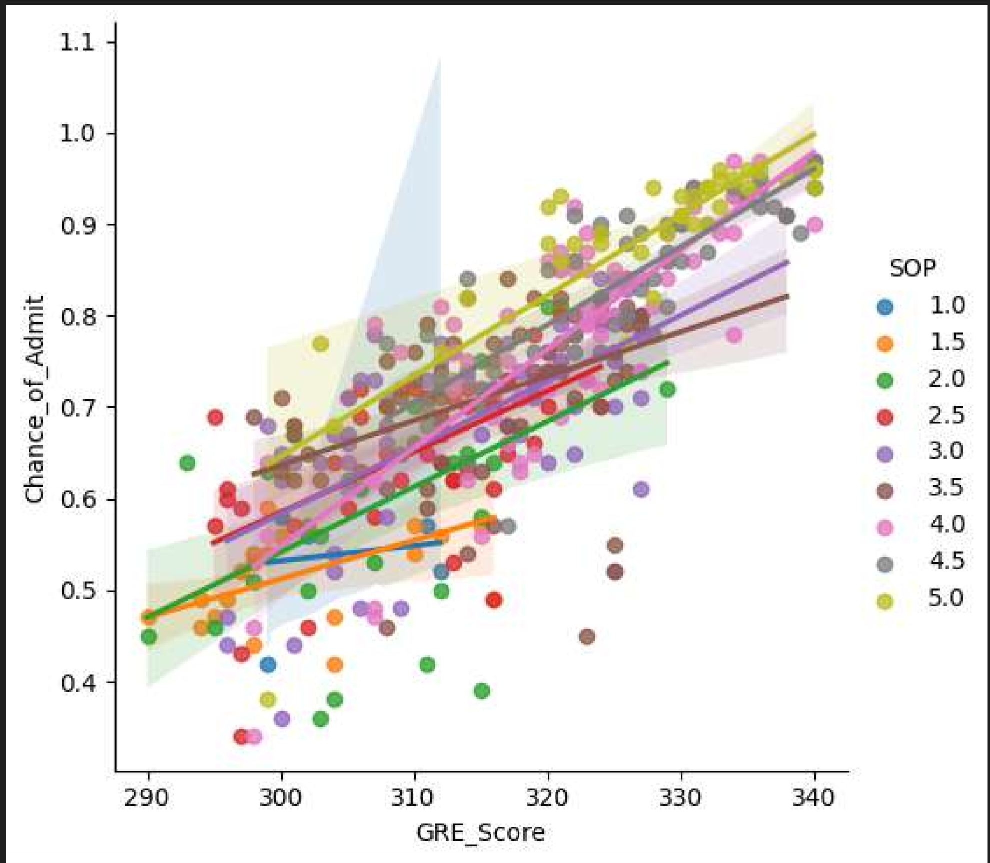
```python
sns.countplot(x=data["Research"])
```

Python

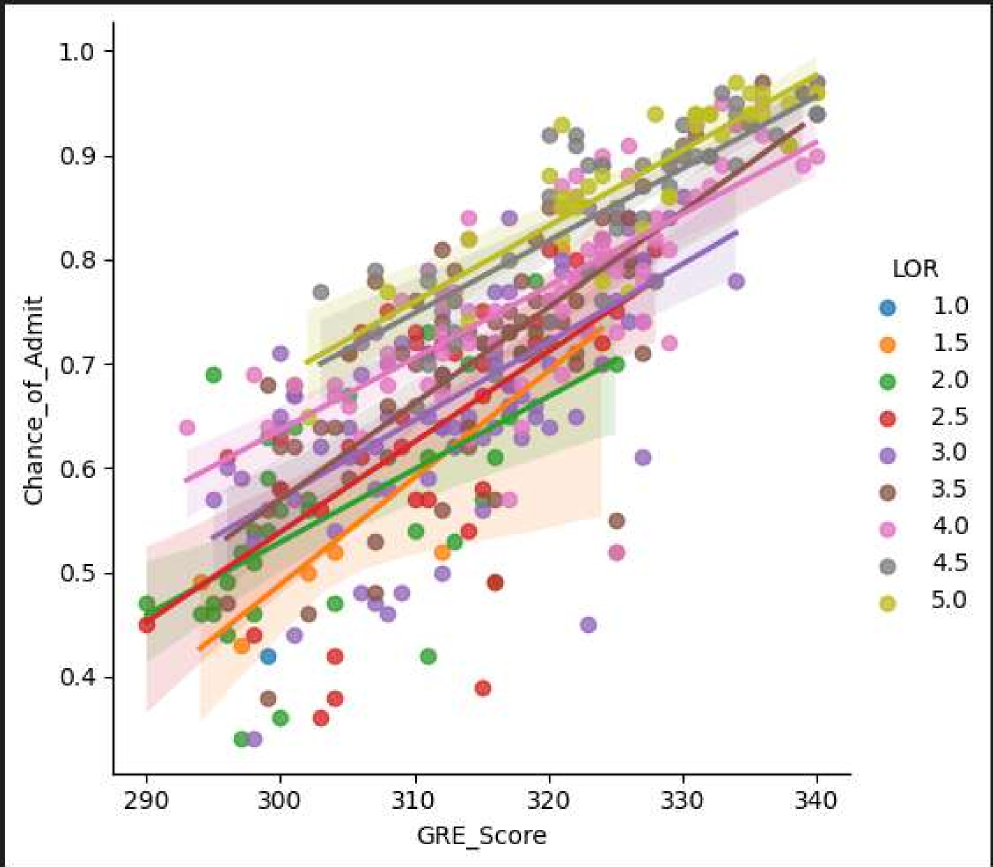
```
<AxesSubplot:xlabel='Research', ylabel='count'>
```



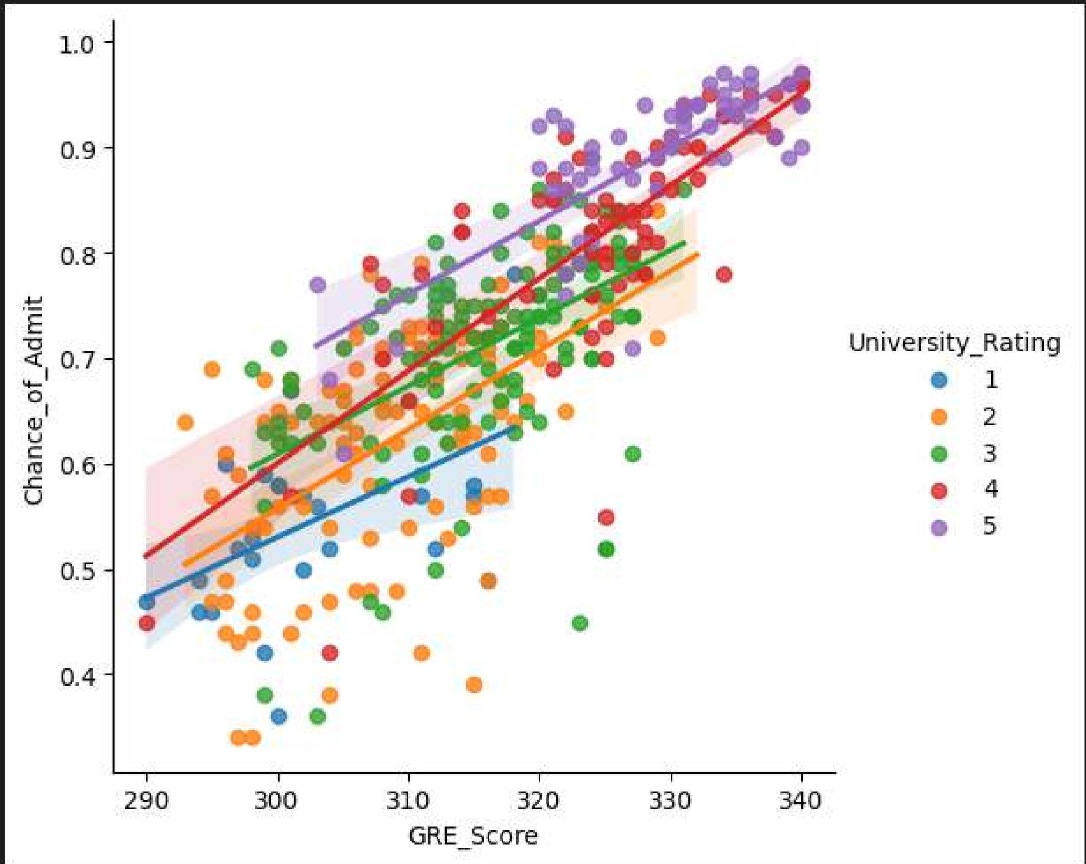

Chance of Admit vs GRE score
Chance_of_Admit
1.0
0.9
0.8
0.7
0.6
0.5
0.4
290
300
310
320
330
340
GRE_Score
Research
0
1



Chance_of_Admit
GRE_Score
SOP
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



Chance_of_Admit
GRE_Score
1.0
0.9
0.8
0.7
0.6
0.5
0.4
290
300
310
320
330
340
LOR
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



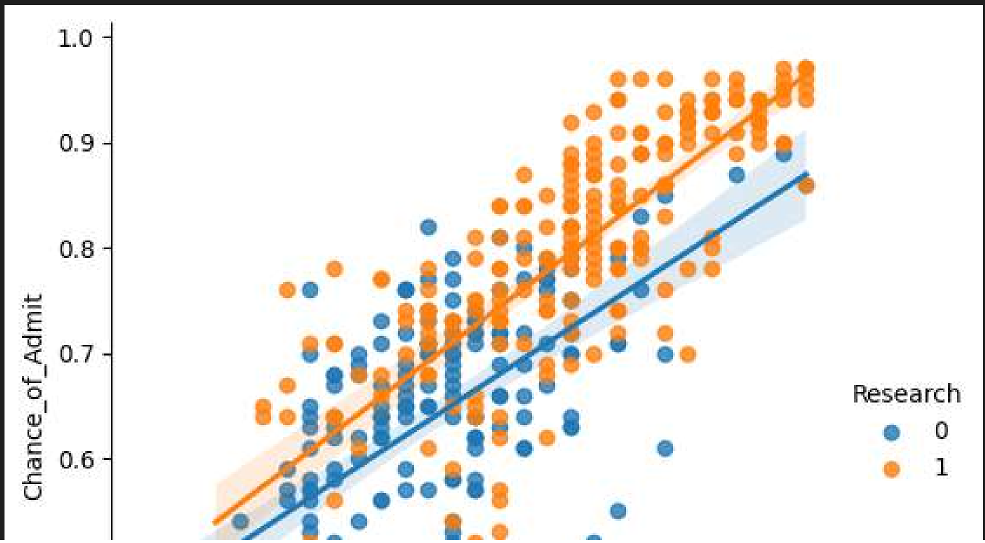
1.0
0.9
0.8
0.7
0.6
0.5
0.4
Chance_of_Admit
290
300
310
320
330
340
GRE_Score
University_Rating
1
2
3
4
5

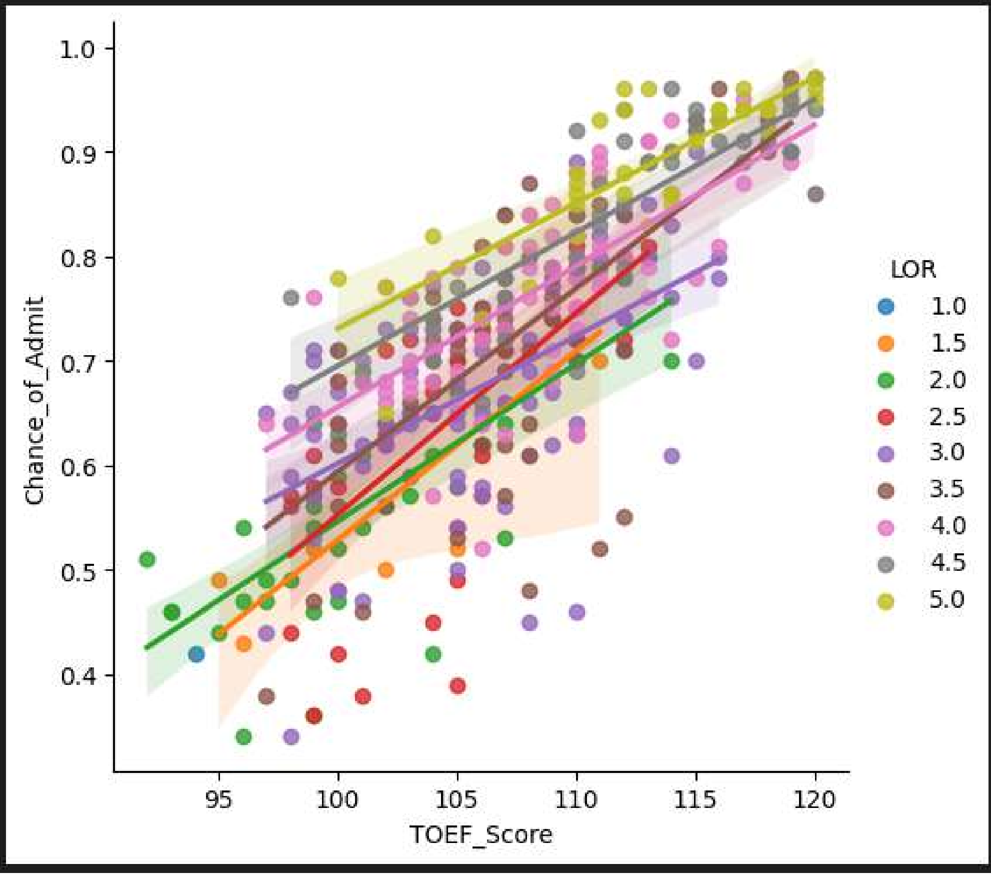
Regresión lineal simple TOEFL Score observando el comportamiento de la variable Research

```python
sns.lmplot(x='TOEF_Score',y='Chance_of_Admit',data=data ,hue='Research')
sns.lmplot(x='TOEF_Score',y='Chance_of_Admit',data=data ,hue='LOR')
sns.lmplot(x='TOEF_Score',y='Chance_of_Admit',data=data ,hue='SOP')
sns.lmplot(x='TOEF_Score',y='Chance_of_Admit',data=data ,hue='University_Rating')
plt.title('Chance of Admit vs TOEFL score')
```

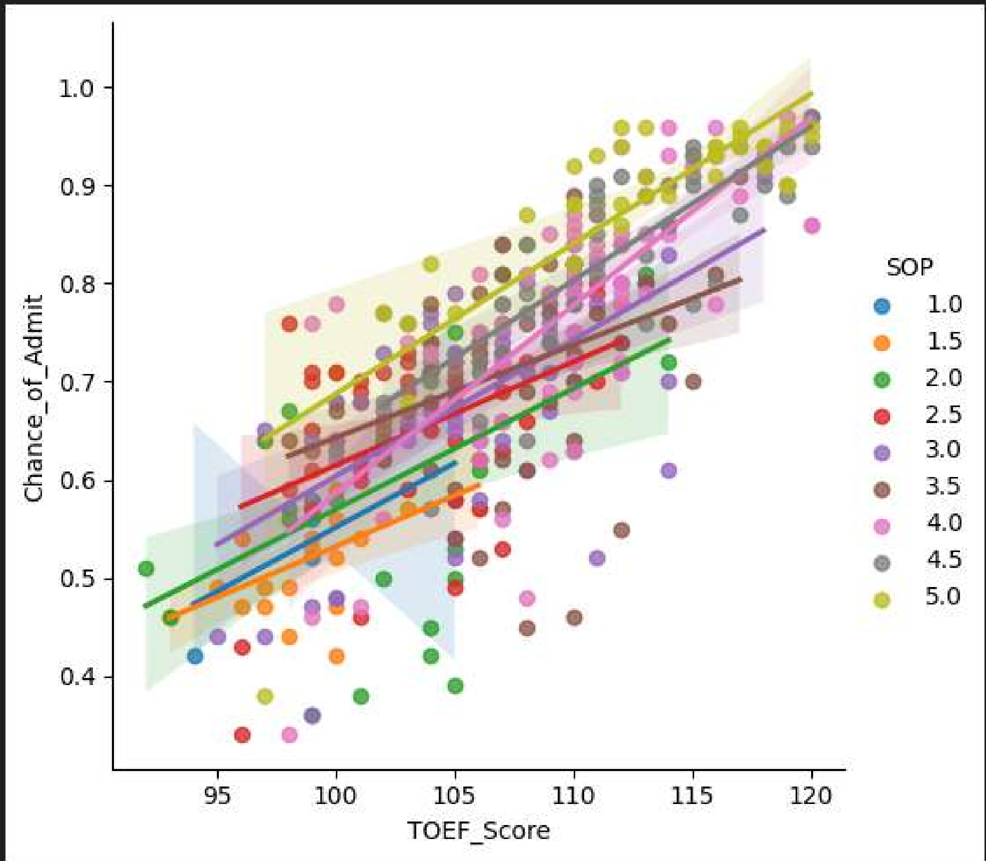
Python

```
Text(0.5, 1.0, 'Chance of Admit vs TOEFL score')
```

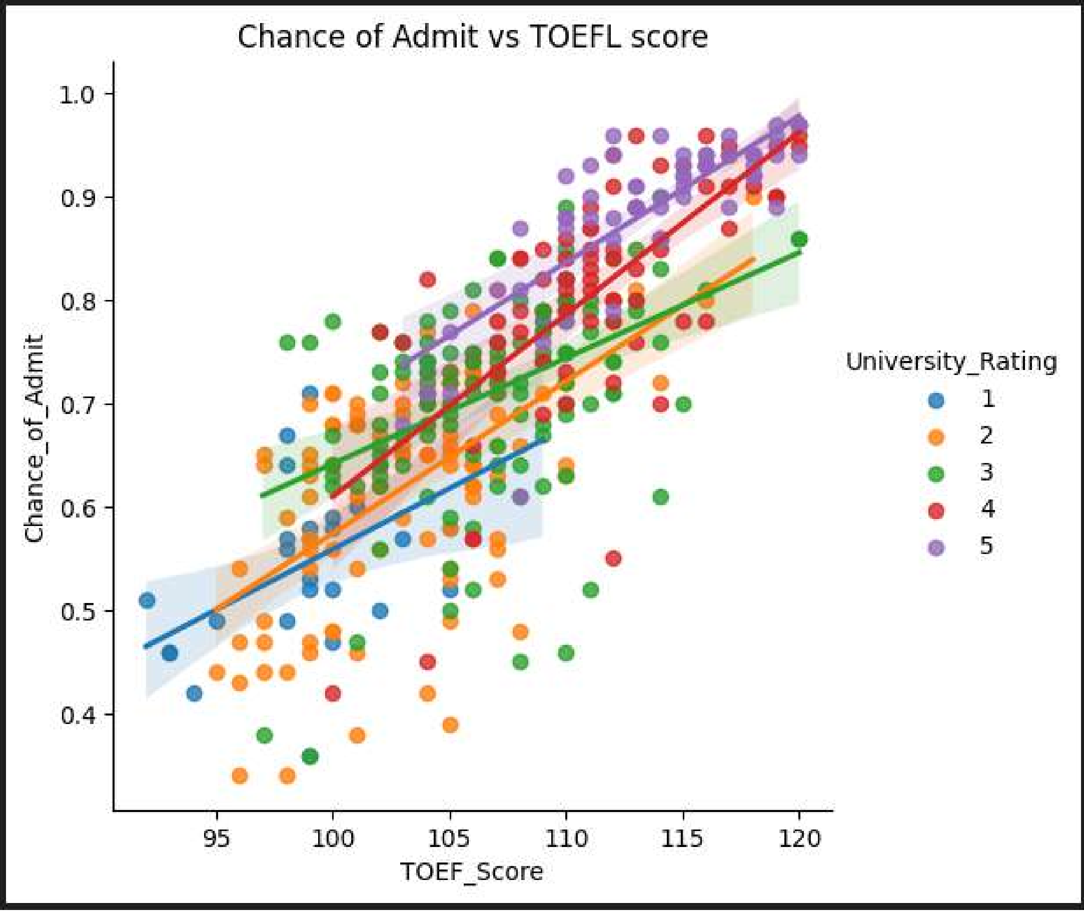




Chance_of_Admit
TOEF_Score
1.0
0.9
0.8
0.7
0.6
0.5
0.4
95
100
105
110
115
120
LOR
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



Chance_of_Admit
TOEF_Score
SOP
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

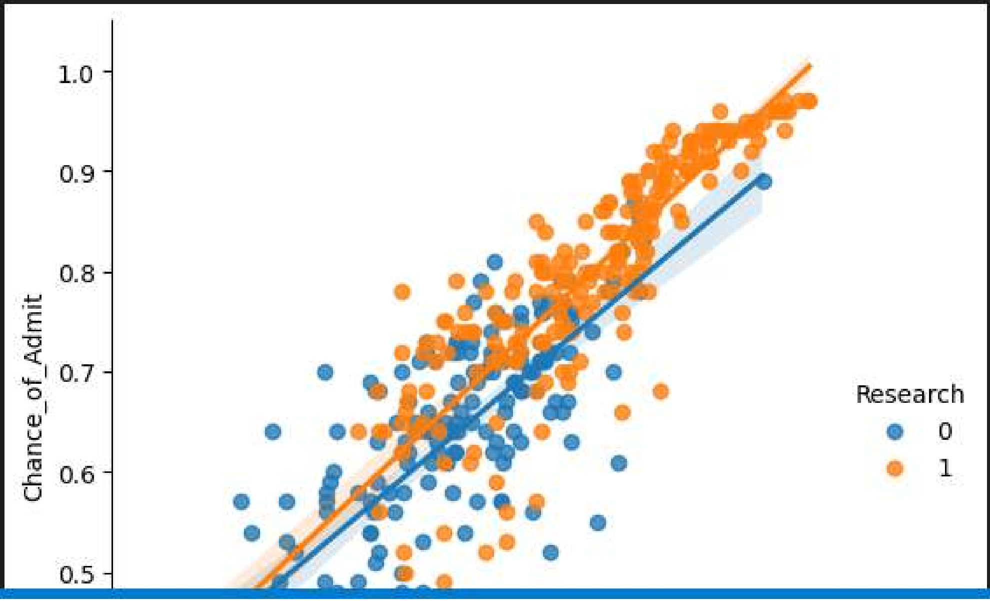

Chance of Admit vs TOEFL score
Chance_of_Admit
TOEF_Score
1.0
0.9
0.8
0.7
0.6
0.5
0.4
95
100
105
110
115
120
University_Rating
1
2
3
4
5

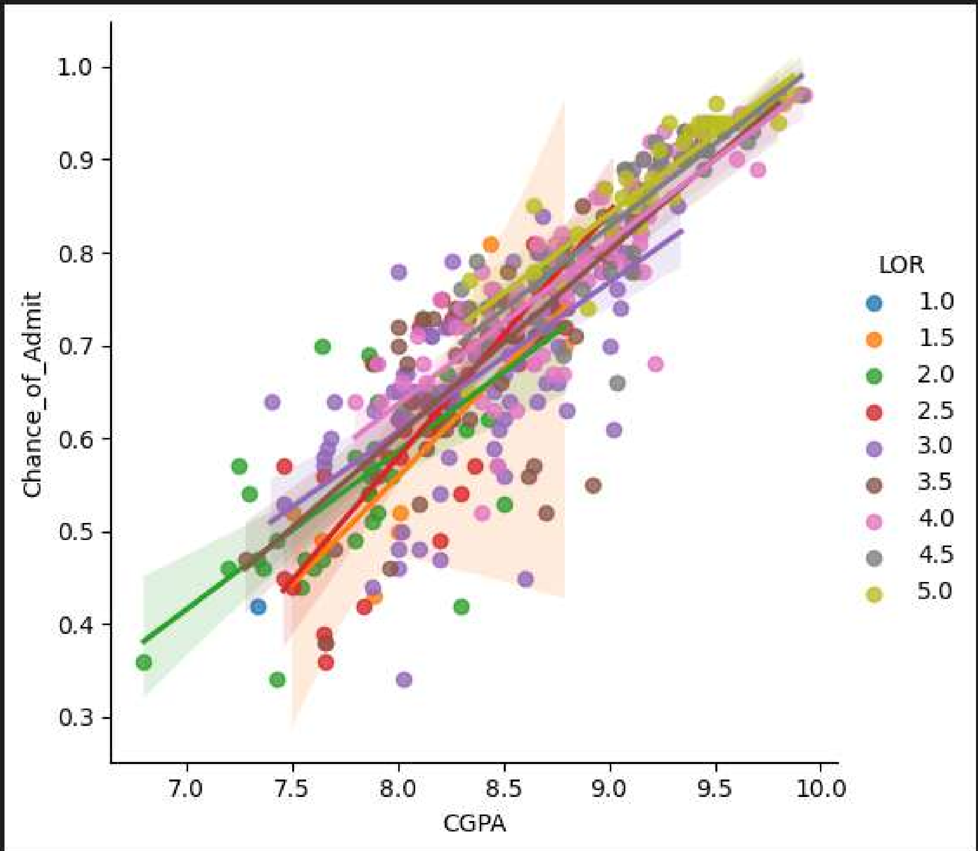
Regresión lineal simple CGPA observando el comportamiento de la variable Research

```python
sns.lmplot(x='CGPA',y='Chance_of_Admit',data=data ,hue='Research')
sns.lmplot(x='CGPA',y='Chance_of_Admit',data=data ,hue='LOR')
sns.lmplot(x='CGPA',y='Chance_of_Admit',data=data ,hue='SOP')
sns.lmplot(x='CGPA',y='Chance_of_Admit',data=data ,hue='University_Rating')
plt.title('Chance of Admit vs CGPA score')
```

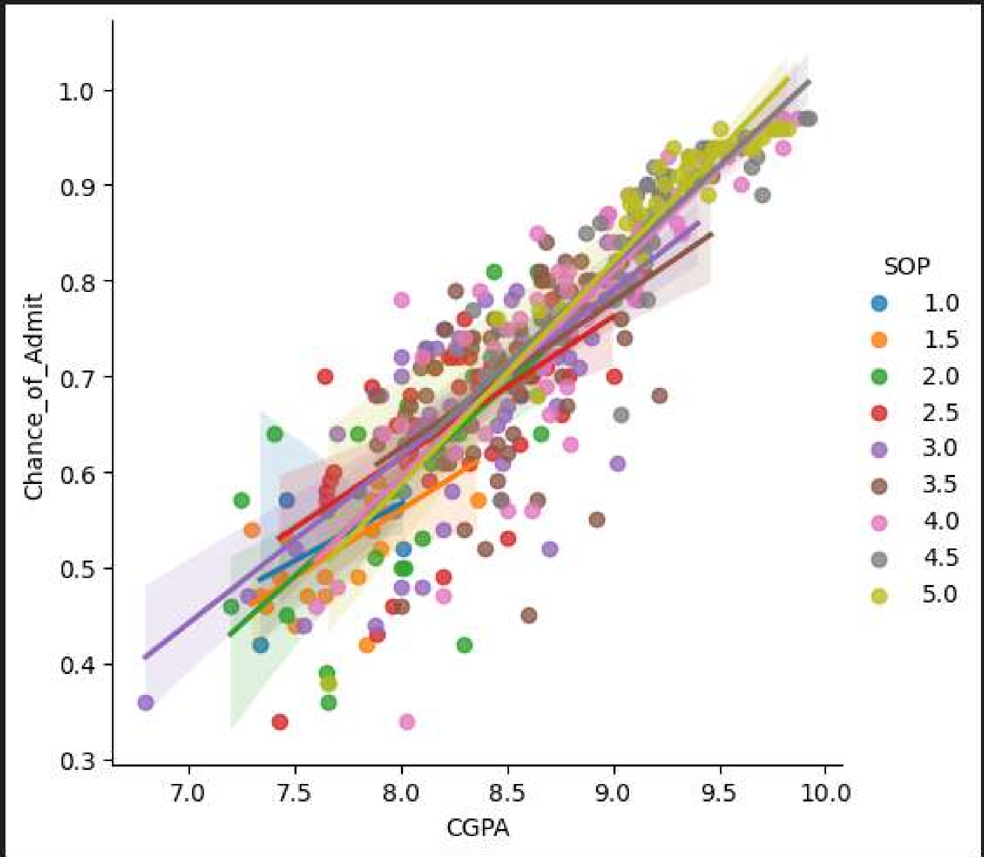
Python

```
Text(0.5, 1.0, 'Chance of Admit vs CGPA score')
```

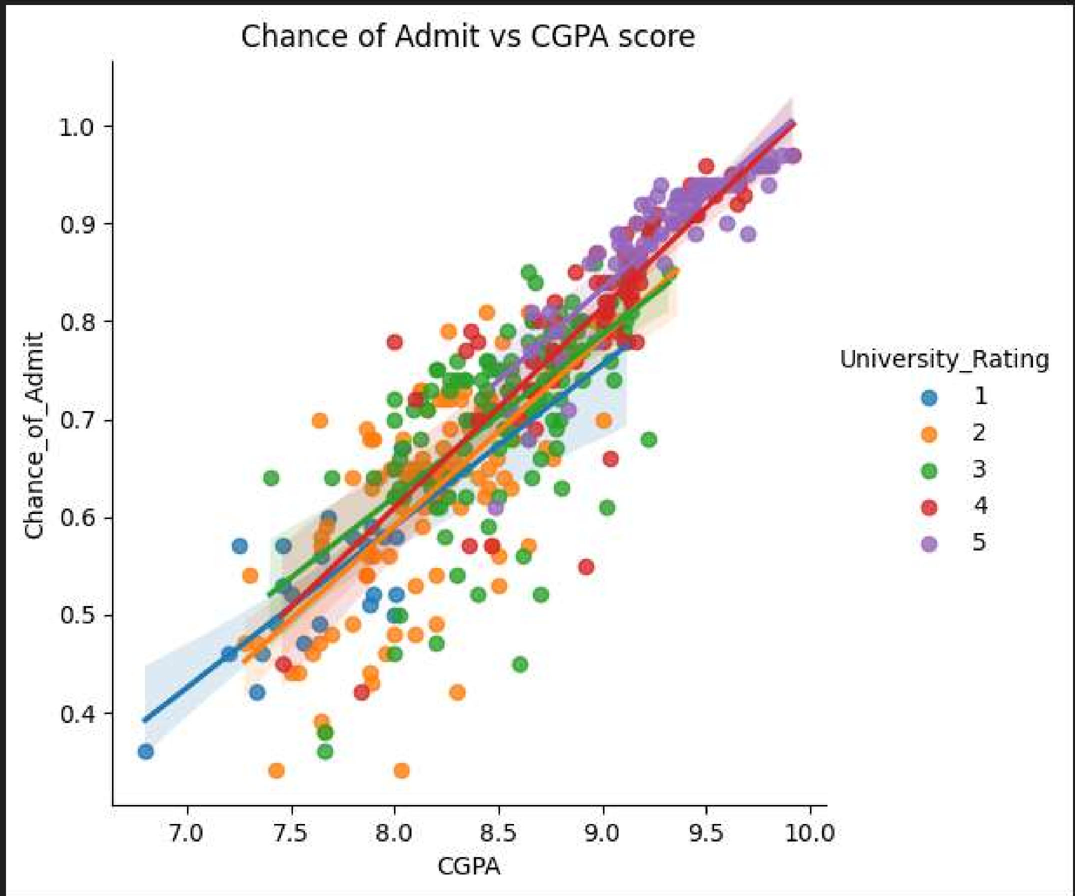




1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
Chance_of_Admit
7.0
7.5
8.0
8.5
9.0
9.5
10.0
CGPA
LOR
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



Chance_of_Admit
CGPA
SOP
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



Chance of Admit vs CGPA score
Chance_of_Admit
CGPA
University_Rating
1
2
3
4
5
1.0
0.9
0.8
0.7
0.6
0.5
0.4
7.0
7.5
8.0
8.5
9.0
9.5
10.0

Conclusiones de lo visualizado

- Desde el mapa de calor, vemos la correlación entre todos los parámetros y, en el caso de “Research” no tiene una correlación lineal con la variable a predecir, esto no quiere decir que se sacara la variable en el modelo, solo se observara de otra manera.
- De los plots para cada una de las variables nos damos cuenta que “University Rating”, “LOR” y “SOP” pueden considerarse como variables categóricas, sin embargo vemos una correlación lineal de estas variables contra la variable a predecir y se comparara los modelos categoricos y los no categoricos para cada variables.
- Viendo las regresiones lineales simple podemos ver los comportamientos que tienen las variables categoricas con cada una de las variables y se puede observar que en las variables University Rating y SOP hay mas disperisión en los datos.

Análisis Predictivo

Vamos a tomar en cuenta el comportamiento de la variable research como variable categórica y de las demás variables categoría se hará una comparación entre tomarla como parte del modelo o como categórica y comparar los resultados

Modelo base

Como ya hemos visto en el mapa de calor, existen tres variables con una fuerte correlacion lineal, es por eso que se tomo la decisión de hacer una regresión lineal multiple con esas tres variables predictoras

```python
a = np.random.randn(len(data))

y_predic = "Chance_of_Admit"
x_columns_base = ["GRE_Score", "TOEF_Score", "CGPA"]
modelo = ["Intercep","GRE_Score", "TOEF_Score", "CGPA"]

data_base = pd.read_csv(r"C:\Users\guill\Desktop\p\MC\Temas\tareas\adm_data.csv", index_col= 0)

check = (a < .8)
entrenamiento = data_base[check]
testeo = data_base[~check]

y_train = entrenamiento[["Chance_of_Admit"]]
x_train_base = entrenamiento[["GRE_Score", "TOEF_Score", "CGPA"]]

entrenamiento
```

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
Serial No.								
1	337	118	4	4.5	4.5	9.65	1	0.92
2	324	107	4	4.0	4.5	8.87	1	0.76
3	316	104	3	3.0	3.5	8.00	1	0.72
4	322	110	3	3.5	2.5	8.67	1	0.80
5	314	103	2	2.0	3.0	8.21	0	0.65

```python
print(list(zip(modelo, regresion_lineal(entrenamiento, x_columns_base,y_predic))))
print("El modelo de regresión lineal queda de la forma:")
display(Math(r'Chance \ of \ Admit  = - 1.657397 + .002423 * GRE \ Score + .001858 * TOEFL \ Score + .156455 * CGPA'))
```

Python

```
[('Intercep', -1.589508582988401), ('GRE_Score', 0.002422569176445677), ('TOEF_Score', 0.003022361963227027), ('CGPA', 0.14193919290652346)]
El modelo de regresión lineal queda de la forma:
```

$$Chance \ of \ Admit = -1.657397 + .002423 * GRE \ Score + .001858 * TOEFL \ Score + .156455 * CGPA$$

```python
lm = LinearRegression()
lm.fit(x_train_base,y_train)
modelo_s = {}

modelo_s = pd.DataFrame(lm.coef_ , columns = x_train_base.columns.values.tolist())
modelo_s["b0"] = lm.intercept_
print("r^2: ",lm.score(x_train_base,y_train))
modelo_s
```

Python

```
r^2:  0.7878947617009404
```

	GRE_Score	TOEF_Score	CGPA	b0
0	0.002423	0.003022	0.141939	-1.589509

Error

+ Code + Markdown

```python
aux = 0
aux = -1.601422 + (0.002401 * entrenamiento["GRE_Score"]) + (0.003053 * entrenamiento["TOEF_Score"]) + (0.143908 * entrenamiento["CGPA"] )
aux.tolist()

SSD = sum((entrenamiento["Chance_of_Admit"] - aux)**2)

RSE = np.sqrt(SSD / (len (entrenamiento) - 4 ))

chance_promedio = np.mean(entrenamiento["Chance_of_Admit"])

error = RSE / chance_promedio

error
```

[22] Python

```
0.09196936671786106
```

Prueba de Hipotesis

```python
lm_b = smf.ols(formula = "Chance_of_Admit~GRE_Score+TOEF_Score+CGPA", data= entrenamiento).fit()
lm_b.summary()
```

[23] Python

OLS Regression Results

Dep. Variable:	Chance_of_Admit	R-squared:	0.788
Model:	OLS	Adj. R-squared:	0.786
Method:	Least Squares	F-statistic:	376.4
Date:	Mon, 28 Nov 2022	Prob (F-statistic):	5.36e-102
Time:	16:08:49	Log-Likelihood:	399.71
No. Observations:	308	AIC:	-791.4
Df Residuals:	304	BIC:	-776.5
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.5895	0.121	-13.132	0.000	-1.828	-1.351
GRE_Score	0.0024	0.001	3.551	0.000	0.001	0.004
TOEF_Score	0.0030	0.001	2.377	0.018	0.001	0.006
CGPA	0.1419	0.012	11.409	0.000	0.117	0.166

Omnibus:	54.747	Durbin-Watson:	1.078
Prob(Omnibus):	0.000	Jarque-Bera (JB):	91.396
Skew:	-1.014	Prob(JB):	1.42e-20
Kurtosis:	4.734	Cond. No.	1.07e+04

Modelo de regresión lineal multiple base agregando la variable Research

```python
columnas = data.columns.values.tolist()
dummy_research = pd.get_dummies(data["Research"], prefix = "research")
data_research = data[columnas].join(dummy_research)
data_research
```

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	research_0	research_1
Serial No.										
1	337	118	4	4.5	4.5	9.65	1	0.92	0	1
2	324	107	4	4.0	4.5	8.87	1	0.76	0	1
3	316	104	3	3.0	3.5	8.00	1	0.72	0	1
4	322	110	3	3.5	2.5	8.67	1	0.80	0	1
5	314	103	2	2.0	3.0	8.21	0	0.65	1	0
...	...	...	...	...	...	...	...	...	...	...
396	324	110	3	3.5	3.5	9.04	1	0.82	0	1
397	325	107	3	3.0	3.5	9.11	1	0.84	0	1
398	330	116	4	5.0	4.5	9.45	1	0.91	0	1
399	312	103	3	3.5	4.0	8.78	0	0.67	1	0
400	333	117	4	5.0	4.0	9.66	1	0.95	0	1

```python
columnas = data_research.columns.values.tolist()

check = (a < .8)
entrenamiento_r = data_research[check]
testeo_r = data_research[~check]

x_columnas_research = ["GRE_Score", "TOEF_Score", "CGPA", "Research"]
x_columnas_research_c = ["GRE_Score", "TOEF_Score", "CGPA", "research_0","research_1"]

x_train_base = entrenamiento_r[["GRE_Score", "TOEF_Score", "CGPA", "Research"]]
x_train_base2c = entrenamiento_r[["GRE_Score", "TOEF_Score", "CGPA", "research_0","research_1"]]
y_train2 = entrenamiento_r[["Chance_of_Admit"]]

modelo_research = ["interception","GRE_Score", "TOEF_Score", "CGPA", "Research"]
modelo_research_c = ["interception","GRE_Score", "TOEF_Score", "CGPA", "research_0","research_1"]
entrenamiento_r
```

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	research_0	research_1
Serial No.										
1	337	118	4	4.5	4.5	9.65	1	0.92	0	1
2	324	107	4	4.0	4.5	8.87	1	0.76	0	1

Modelo No categorico

```python
print(list(zip(modelo_research, regresion_lineal(entrenamiento_r, x_columnas_research,y_predic))))
```

```
[('interception', -1.4105026746366047), ('GRE_Score', 0.0018642173781217544), ('TOEF_Score', 0.003043653351462074), ('CGPA', 0.13981407245372832), ('Research', 0.026292617902504884)]
```

```python
lm_r = LinearRegression()
lm_r.fit(x_train_base,y_train2)
modelo_dresearch = {}

modelo_dresearch = pd.DataFrame(lm_r.coef_ , columns = x_train_base.columns.values.tolist())
modelo_dresearch["b0"] = lm_r.intercept_

display(Math(r'Chance \ of \ Admit = -1.502683 + 0.002185 * GRE \ Score + 0.003034 * TOEFL \ Score + 0.139622 * CGPA + 0.0203 * Research'))
modelo_dresearch
```

$$Chance \ of \ Admit = -1.502683 + 0.002185 * GRE \ Score + 0.003034 * TOEFL \ Score + 0.139622 * CGPA + 0.0203 * Research$$

	GRE_Score	TOEF_Score	CGPA	Research	b0
0	0.001864	0.003044	0.139814	0.026293	-1.410503

Modelo Categorico

```python
print(list(zip(modelo_research_c, regresion_lineal(entrenamiento_r, x_columnas_research_c,y_predic))))
```

[28] Python

```
[('interception', -39.0), ('GRE_Score', 0.11001781419410861), ('TOEF_Score', 0.008488729558344232), ('CGPA', -0.993661527644548), ('research_0', 13.0),
('research_1', 13.0)]
```

```python
lm_rc = LinearRegression()
lm_rc.fit(x_train_base2c,y_train)

modelo_dresearch_c = {}
modelo_dresearch_c = pd.DataFrame(lm_rc.coef_, columns = x_train_base2c.columns.values.tolist())
modelo_dresearch_c["b0"] = lm_rc.intercept_


display(Math(r'Chance \ of \ Admit = -1.267598 + 0.00136 * GRE \ Score + 0.002863 * TOEFL \ Score + 0.145653 * CGPA -0.015822 * research_0 + 0.015822 * researc
modelo_dresearch_c
```

[29] Python

$$Chance\ of\ Admit = -1.267598 + 0.00136 * GRE\ Score + 0.002863 * TOEFL\ Score + 0.145653 * CGPA - 0.015822 * research_0 + 0.015822 * research_1$$

	GRE_Score	TOEF_Score	CGPA	research_0	research_1	b0
0	0.001864	0.003044	0.139814	-0.013146	0.013146	-1.397356

Comparacion de r^2 de ambos modelos

```python
print("No categorica: ",lm_r.score(x_train_base,y_train),"\n Categorica: ",lm_rc.score(x_train_base2c,y_train))
```

Python

```
No categorica:  0.7935469993130141
 Categorica:  0.7935469993130141
```

Tenemos que la diferencia de las r^2 no es significativaa por eso se tomara el modelo más simple

Modelo de regresión lineal multiple base agregando la variable University rating

```python
columnas = data.columns.values.tolist()
dummy_rating = pd.get_dummies(data["University_Rating"], prefix = "Rating")
data_rating = data[columnas].join(dummy_rating)
data_rating
```

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	Rating_1	Rating_2	Rating_3	Rating_4	Rating_5
Serial No.													
1	337	118	4	4.5	4.5	9.65	1	0.92	0	0	0	1	0

```python
check = (a < .8)
entrenamiento_rating = data_rating[check]
testeo_rating = data_rating[~check]

x_columnas_rating = ["GRE_Score", "TOEF_Score", "CGPA", "University_Rating"]
x_columnas_rating_c = ["GRE_Score", "TOEF_Score", "CGPA", "Rating_1", "Rating_2","Rating_3","Rating_4","Rating_5"]

x_train_base_rating = entrenamiento_rating[["GRE_Score", "TOEF_Score", "CGPA", "University_Rating"]]
x_train_base_rating_c = entrenamiento_rating[["GRE_Score", "TOEF_Score", "CGPA", "Rating_1", "Rating_2","Rating_3","Rating_4","Rating_5"]]
y_train_rating = entrenamiento_rating[["Chance_of_Admit"]]

modelo_rating=  ["Intercept","GRE_Score", "TOEF_Score", "CGPA", "University_Rating"]
modelo_rating_c = ["Intercept","GRE_Score", "TOEF_Score", "CGPA", "Rating_1", "Rating_2","Rating_3","Rating_4","Rating_5"]
entrenamiento_rating
```

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	Rating_1	Rating_2	Rating_3	Rating_4	Rating_5
Serial No.													
1	337	118	4	4.5	4.5	9.65	1	0.92	0	0	0	1	0
2	324	107	4	4.0	4.5	8.87	1	0.76	0	0	0	1	0
3	316	104	3	3.0	3.5	8.00	1	0.72	0	0	1	0	0
4	322	110	3	3.5	2.5	8.67	1	0.80	0	0	1	0	0
5	314	103	2	2.0	3.0	8.21	0	0.65	0	1	0	0	0

Modelo sin categorico

```python
print(list(zip(modelo_rating, regresion_lineal(entrenamiento_rating, x_columnas_rating,y_predic))))
```

[33] Python

```
[('Intercept', -1.4848763940629905), ('GRE_Score', 0.0023315223365292775), ('TOEF_Score', 0.0026995669085706805), ('CGPA', 0.13415223921847522),
('University_Rating', 0.008332446224552514)]
```

```python
lm_rating = LinearRegression()
lm_rating.fit(x_train_base_rating,y_train_rating)
modelo_drating = {}

modelo_drating = pd.DataFrame(lm_rating.coef_ , columns = x_train_base_rating.columns.values.tolist())
modelo_drating["b0"] = lm_rating.intercept_

display(Math(r'Chance \ of \ Admit = -1.508599 + 0.002567 * GRE \ Score + 0.002522 * TOEFL \ Score + 0.129749 * CGPA + 0.011617 * University \ Rating'))
modelo_drating
```

[34] Python

$$Chance \ of \ Admit = -1.508599 + 0.002567 * GRE \ Score + 0.002522 * TOEFL \ Score + 0.129749 * CGPA + 0.011617 * University \ Rating$$

	GRE_Score	TOEF_Score	CGPA	University_Rating	b0
0	0.002332	0.0027	0.134152	0.008332	-1.484876

Modelo categorico

```
print(list(zip(modelo_rating_c, regresion_lineal(entrenamiento_rating, x_columnas_rating_c,y_predic))))
```

```
[('Intercept', 61.43453124999999), ('GRE_Score', -0.1954514468092899), ('TOEF_Score', 0.013724660977417258), ('CGPA', 0.9124719104071617), ('Rating_1', -11.272812500000003), ('Rating_2', -9.721875000000033), ('Rating_3', -8.718281250000032), ('Rating_4', -6.730468750000034), ('Rating_5', -7.049687500000035)]
```

```
lm_rating_c = LinearRegression()
lm_rating_c.fit(x_train_base_rating_c,y_train_rating)
modelo_drating_c = {}

modelo_drating_c = pd.DataFrame(lm_rating_c.coef_ , columns = x_train_base_rating_c.columns.values.tolist())
modelo_drating_c["b0"] = lm_rating_c.intercept_

display(Math(r'Chance \ of \ Admit = -1.467998 + 0.002511 * GRE \ Score + 0.002587 * TOEFL \ Score + 0.130529 * CGPA  -0.013852 * University \ Rating \ 1 -0.01
modelo_drating_c
```

$$Chance\ of\ Admit = -1.467998 + 0.002511 * GRE\ Score + 0.002587 * TOEFL\ Score + 0.130529 * CGPA - 0.013852 * University\ Rating\ 1 - 0.016509 * University\ Rating\ 2 - 0.001764 * University\ Rating\ 3 + 0.009708 * University\ Rating\ 4 + 0.022417 * University\ Rating\ 5$$

	GRE_Score	TOEF_Score	CGPA	Rating_1	Rating_2	Rating_3	Rating_4	Rating_5	b0
0	0.002284	0.002792	0.134941	-0.004781	-0.010247	-0.008349	0.001032	0.022345	-1.458554

Comparacion de R^2

```python
print("No categorica: ",lm_rating.score(x_train_base_rating,y_train),"\n Categorica: ",lm_rating_c.score(x_train_base_rating_c,y_train))
```

[37]

```
No categorica:  0.7895883479099702
 Categorica:  0.7914703872808827
```

Modelo de regresión lineal multiple base agregando la variable LOR

Datos

```python
columnas = data.columns.values.tolist()
dummy_LOR = pd.get_dummies(data["LOR"], prefix = "LOR")
data_lor = data[columnas].join(dummy_LOR)
data_lor
```

[38]

```python
check = (a < .8)
entrenamiento_lor = data_lor[check]
testeo_lor = data_lor[~check]

x_columnas_lor = ["GRE_Score", "TOEF_Score", "CGPA", "LOR"]
x_columnas_lor_c =  ["GRE_Score", "TOEF_Score", "CGPA", "LOR_1.0","LOR_1.5","LOR_2.0","LOR_2.5","LOR_3.0","LOR_3.5","LOR_4.0","LOR_4.5","LOR_5.0"]

x_train_base_lor = entrenamiento_lor[["GRE_Score", "TOEF_Score", "CGPA", "LOR"]]
x_train_base_lor_c = entrenamiento_lor[["GRE_Score", "TOEF_Score", "CGPA", "LOR_1.0","LOR_1.5","LOR_2.0","LOR_2.5","LOR_3.0","LOR_3.5","LOR_4.0","LOR_4.5","LOR
y_train_lor = entrenamiento_lor[["Chance_of_Admit"]]

modelo_lor=  ["Intercept","GRE_Score", "TOEF_Score", "CGPA", "LOR"]
modelo_lor_c = ["Intercept","GRE_Score", "TOEF_Score", "CGPA", "LOR_1.0","LOR_1.5","LOR_2.0","LOR_2.5","LOR_3.0","LOR_3.5","LOR_4.0","LOR_4.5","LOR_5.0"]
entrenamiento_lor
```

9]

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	LOR_1.0	LOR_1.5	LOR_2.0	LOR_2.5	LOR_3.0	LOR_3.5	LOR_4.0	LOR_4.5
Serial No.																
1	337	118	4	4.5	4.5	9.65	1	0.92	0	0	0	0	0	0	0	1
2	324	107	4	4.0	4.5	8.87	1	0.76	0	0	0	0	0	0	0	1
3	316	104	3	3.0	3.5	8.00	1	0.72	0	0	0	0	0	1	0	0
4	322	110	3	3.5	2.5	8.67	1	0.80	0	0	0	1	0	0	0	0
5	314	103	2	2.0	3.0	8.21	0	0.65	0	0	0	0	1	0	0	0

Modelo No categorico

```
print(list(zip(modelo_lor, regresion_lineal(entrenamiento_lor, x_columnas_lor,y_predic))))
```

[40] Python

```
[('Intercept', -1.4664460434940414), ('GRE_Score', 0.0024215745799701276), ('TOEF_Score', 0.0030934034604770443), ('CGPA', 0.1168218654361129), ('LOR',
0.024647206412470472)]
```

```
lm_lor = LinearRegression()
lm_lor.fit(x_train_base_lor,y_train_lor)
modelo_dlor = {}

modelo_dlor = pd.DataFrame(lm_lor.coef_ , columns = x_train_base_lor.columns.values.tolist())
modelo_dlor["b0"] = lm_lor.intercept_

display(Math(r'Chance \ of \ Admit = -1.419068 + 0.002362 * GRE \ Score + 0.002166 * TOEFL \ Score + 0.129795 * CGPA + 0.023613 * LOR'))
modelo_dlor
```

[41] Python

$$Chance \ of \ Admit = -1.419068 + 0.002362 * GRE \ Score + 0.002166 * TOEFL \ Score + 0.129795 * CGPA + 0.023613 * LOR$$

	GRE_Score	TOEF_Score	CGPA	LOR	b0
0	0.002422	0.003093	0.116822	0.024647	-1.466446

Modelo Categorico

```python
print(list(zip(modelo_lor_c, regresion_lineal(entrenamiento_lor, x_columnas_lor_c,y_predic))))
```

[42] Python

```
[('Intercept', 11.012803819444445), ('GRE_Score', -0.12810045964541336), ('TOEF_Score', 0.04141026453588752), ('CGPA', 1.8282516808250462), ('LOR_1.0', 8.869305555555556), ('LOR_1.5', 8.777469618055555), ('LOR_2.0', 8.384149305555555), ('LOR_2.5', 10.380438368055556), ('LOR_3.0', 9.082664930555556), ('LOR_3.5', 7.975985243055556), ('LOR_4.0', 8.270047743055555), ('LOR_4.5', 7.670789930555555), ('LOR_5.0', 8.258914930555555)]
```

```python
lm_lor_c = LinearRegression()
lm_lor_c.fit(x_train_base_lor_c,y_train_lor)
modelo_dlor_c = {}

modelo_dlor_c = pd.DataFrame(lm_lor_c.coef_ , columns = x_train_base_lor_c.columns.values.tolist())
modelo_dlor_c["b0"] = lm_lor_c.intercept_

display(Math(r'Chance \ of \ Admit = -1.420869 + 0.002579 * GRE \ Score + 0.002499* TOEFL \ Score + 0.122007 * CGPA -0.060629 * LOR_{1.0} -0.033728* LOR_{1.5}
modelo_dlor_c
```

[43] Python

$$Chance \ of \ Admit = -1.420869 + 0.002579 * GRE \ Score + 0.002499 * TOEFL \ Score + 0.122007 * CGPA - 0.060629 * LOR_{1.0} - 0.033728 * LOR_{1.5} - 0.013655 * LOR_{2.0} - 0.010285 * LOR_{2.5} - 0.003593 * LOR_{3.0} + 0.007921 * LOR_{3.5} + 0.028188 * LOR_{4.0} + 0.034129 * LOR_{4.5} + 0.051652 * LOR_{5.0}$$

	GRE_Score	TOEF_Score	CGPA	LOR_1.0	LOR_1.5	LOR_2.0	LOR_2.5	LOR_3.0	LOR_3.5	LOR_4.0	LOR_4.5	LOR_5.0	b0
0	0.00237	0.003253	0.116973	-0.059078	-0.032689	-0.014721	-0.006798	-0.009101	0.004501	0.02643	0.036189	0.055266	-1.393946

Comparación de R^2

```python
print("No categorica: ",lm_lor.score(x_train_base_lor,y_train_lor),"\n Categorica: ",lm_lor_c.score(x_train_base_lor_c,y_train_lor))
```

```
No categorica:  0.8018902517538589
 Categorica:  0.8039628752167377
```

Modelo de regresión lineal multiple base agregando la variable SOP

Datos

```python
columnas = data.columns.values.tolist()
dummy_sop = pd.get_dummies(data["SOP"], prefix = "SOP")
data_sop = data[columnas].join(dummy_sop)
data_sop
```

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	SOP_1.0	SOP_1.5	SOP_2.0	SOP_2.5	SOP_3.0	SOP_3.5	SOP_4.0	SOP_4.5
Serial No.																

```python
check = (a < .8)
entrenamiento_sop = data_sop[check]
testeo_sop = data_sop[~check]

x_columnas_sop = ["GRE_Score", "TOEF_Score", "CGPA", "SOP"]
x_columnas_sop_c =  ["GRE_Score", "TOEF_Score", "CGPA", "SOP_1.0","SOP_1.5","SOP_2.0","SOP_2.5","SOP_3.0","SOP_3.5","SOP_4.0","SOP_4.5","SOP_5.0"]

x_train_base_sop = entrenamiento_sop[["GRE_Score", "TOEF_Score", "CGPA", "SOP"]]
x_train_base_sop_c = entrenamiento_sop[["GRE_Score", "TOEF_Score", "CGPA", "SOP_1.0","SOP_1.5","SOP_2.0","SOP_2.5","SOP_3.0","SOP_3.5","SOP_4.0","SOP_4.5","SOP
y_train_sop = entrenamiento_sop[["Chance_of_Admit"]]

modelo_sop=  ["Intercept","GRE_Score", "TOEF_Score", "CGPA", "SOP"]
modelo_sop_c = ["Intercept","GRE_Score", "TOEF_Score", "CGPA", "SOP_1.0","SOP_1.5","SOP_2.0","SOP_2.5","SOP_3.0","SOP_3.5","SOP_4.0","SOP_4.5","SOP_5.0"]
entrenamiento_sop
```

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit	SOP_1.0	SOP_1.5	SOP_2.0	SOP_2.5	SOP_3.0	SOP_3.5	SOP_4.0	SOP_4.5
Serial No.																
1	337	118	4	4.5	4.5	9.65	1	0.92	0	0	0	0	0	0	0	1
2	324	107	4	4.0	4.5	8.87	1	0.76	0	0	0	0	0	0	1	0
3	316	104	3	3.0	3.5	8.00	1	0.72	0	0	0	0	1	0	0	0
4	322	110	3	3.5	2.5	8.67	1	0.80	0	0	0	0	0	1	0	0
5	314	103	2	2.0	3.0	8.21	0	0.65	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	329	111	4	4.5	4.0	9.23	1	0.89	0	0	0	0	0	0	0	1

Modelo no categorico

```
print(list(zip(modelo_sop, regresion_lineal(entrenamiento_sop, x_columnas_sop,y_predic))))
```

[47] Python

```
[('Intercept', -1.4915613406786288), ('GRE_Score', 0.0024831715210344722), ('TOEF_Score', 0.002465557755704353), ('CGPA', 0.1301529784412378), ('SOP', 0.012943660788073 31)]
```

```
lm_sop = LinearRegression()
lm_sop.fit(x_train_base_sop,y_train_sop)
modelo_dsop = {}

modelo_dsop = pd.DataFrame(lm_sop.coef_ , columns = x_train_base_sop.columns.values.tolist())
modelo_dsop["b0"] = lm_sop.intercept_

display(Math(r'Chance \ of \ Admit = -1.503781 + 0.002533 * GRE \ Score + 0.002286 * TOEFL \ Score + 0.132345 * CGPA + 0.012571 * SOP'))
modelo_dsop
```

[48] Python

$$Chance \ of \ Admit = -1.503781 + 0.002533 * GRE \ Score + 0.002286 * TOEFL \ Score + 0.132345 * CGPA + 0.012571 * SOP$$

	GRE_Score	TOEF_Score	CGPA	SOP	b0
0	0.002483	0.002466	0.130153	0.012944	-1.491561

Modelo Categorico

```python
print(list(zip(modelo_sop_c, regresion_lineal(entrenamiento_sop, x_columnas_sop_c,y_predic))))
```

[49] Python

```
[('Intercept', 134.73000000000002), ('GRE_Score', -0.5807736225324541), ('TOEF_Score', -0.042188912615748045), ('CGPA', 2.8573295267532584), ('SOP_1.0', 41.99249999999997), ('SOP_1.5', 34.82249999999998), ('SOP_2.0', 22.897499999999976), ('SOP_2.5', 28.519999999999975), ('SOP_3.0', 44.519999999999975), ('SOP_3.5', 43.00999999999998), ('SOP_4.0', 52.59499999999998), ('SOP_4.5', 49.424999999999976), ('SOP_5.0', 44.59499999999998)]
```

```python
lm_sop_c = LinearRegression()
lm_sop_c.fit(x_train_base_sop_c,y_train_sop)
modelo_dsop_c = {}

modelo_dsop_c = pd.DataFrame(lm_sop_c.coef_ , columns = x_train_base_sop_c.columns.values.tolist())
modelo_dsop_c["b0"] = lm_sop_c.intercept_

display(Math(r'Chance \ of \ Admit = -1.478403 + 0.002559 * GRE \ Score + 0.002516 * TOEFL \ Score + 0.130189 * CGPA + -0.007071 * SOP_{1.0} -0.02113* SOP_{1.5
modelo_dsop_c
```

[50] Python

$$Chance\ of\ Admit = -1.478403 + 0.002559 * GRE\ Score + 0.002516 * TOEFL\ Score + 0.130189 * CGPA + -0.007071 * SOP_{1.0} - 0.02113 * SOP_{1.5} - 0.024223 * SOP_{2.0} - 0.002659 * SOP_{2.5} + 0.005314 * SOP_{3.0} - 0.002585 * SOP_{3.5} + 0.003638 * SOP_{4.0} + 0.010822 * SOP_{4.5} + 0.037895 * SOP_{5.0}$$

	GRE_Score	TOEF_Score	CGPA	SOP_1.0	SOP_1.5	SOP_2.0	SOP_2.5	SOP_3.0	SOP_3.5	SOP_4.0	SOP_4.5	SOP_5.0	b0
0	0.002545	0.002647	0.127984	-0.01878	-0.033466	-0.016023	0.009687	0.001743	-0.00246	0.008799	0.013271	0.037227	-1.472321

Resumen

Se pudo observar que en todas al ser variables numericas vimos como las pruebas r^2 no se dispersaron mucho, ademas al tener un Dataset muy pequeño se opto por tratarlas como variables normales, en lugar de categoricas para poder tener un mejor modelo

Analisis Predictivo

En el punto anterior pudimos observar que apartir de un modelo base, como se comportan las variables categoricas agrupandolas y no agrupandolas y se hizo la pruba r cuadrada para cada caso y no se encontro mejorias significativas para no agruparlas, es por eso que en esta sección se considerara una regresión lineal multiple con todas las variables sin agrupación y se haran las prubas para determinar si es la mejor opción

Regresión lineal multiple con todas las variables como predictoras

Datos

```python
x_columns_mult = ["GRE_Score", "TOEF_Score", "CGPA", "SOP", "LOR", "University_Rating", "Research"]

data_base = pd.read_csv(r"C:\Users\guill\Desktop\p\MC\Temas\tareas\adm_data.csv", index_col= 0)

check = (a < .8)
entrenamiento_m = data_base[check]
testeo_m = data_base[~check]

modelo_mult = ["Intercept" , "GRE_Score", "TOEF_Score", "CGPA", "SOP", "LOR", "University_Rating", "Research"]

y_train_m = entrenamiento_m[["Chance_of_Admit"]]
x_train_m = entrenamiento_m[["GRE_Score", "TOEF_Score", "CGPA", "SOP", "LOR", "University_Rating", "Research"]]

entrenamiento_m
```

[52] Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
Serial No.								
1	337	118	4	4.5	4.5	9.65	1	0.92
2	324	107	4	4.0	4.5	8.87	1	0.76
3	316	104	3	3.0	3.5	8.00	1	0.72
4	322	110	3	3.5	2.5	8.67	1	0.80

Modelo

```python
print(list(zip(modelo_mult, regresion_lineal(entrenamiento_m, x_columns_mult, y_predic))))
```

Python

```
[('Intercept', -1.3281559240501792), ('GRE_Score', 0.0019536964837225668), ('TOEF_Score', 0.0031343388175155695), ('CGPA', 0.11672284536772537), ('SOP', 0.0006106277353343792), ('LOR', 0.023609544054325582), ('University_Rating', -0.0013341931350598912), ('Research', 0.0228553324838109977)]
```

+ Code + Markdown

```python
lm_m = LinearRegression()
lm_m.fit(x_train_m,y_train_m)
modelo_m = {}

modelo_m = pd.DataFrame(lm_m.coef_ , columns = x_train_m.columns.values.tolist())
modelo_m["b0"] = lm_m.intercept_

print("El modelo de regresión lineal queda de la forma:")
display(Math(r'Chance \ of \ Admit  =   -1.244028 + 0.001903 * GRE \ Score + 0.002438 * TOEFL \ Score + 0.116411 * CGPA - -0.001317 * SOP + 0.0191 LOR + 0.0094

print("r^2: ",lm_m.score(x_train_m, y_train_m))



modelo_m
```

Python

```
El modelo de regresión lineal queda de la forma:
```

$$Chance\ of\ Admit = -1.244028 + 0.001903 * GRE\ Score + 0.002438 * TOEFL\ Score + 0.116411 * CGPA - -0.001317 * SOP + 0.0191 LOR + 0.009468 * University\ Rating + 0.020616 * Research$$

```
r^2:  0.8061341096457737
```

	GRE_Score	TOEF_Score	CGPA	SOP	LOR	University_Rating	Research	b0
0	0.001954	0.003134	0.116723	0.000611	0.02361	-0.001334	0.022855	-1.328156

Aquí se tiene un problema, al tener un modelo mas complejo, se esperaria que el valor de R^2 aumentase sin embargo el incremento en el valor es insignificante o no es lo esperado al modelo base asi que se se le sometera a unas pruebas y observar que pasa

VIF

- GRE = TOEF + CGPA + UR + SOP + LOR + Research
- TOEF = GRE + CGPA + UR + SOP + LOR + Research
- CGPA = GRE + TOEF + + UR + SOP + LOR + Research
- UR = GRE + TOEF + CGPA + SOP + LOR + Research
- SOP = GRE + TOEF + CGPA + UR + LOR + Research
- LOR = GRE + TOEF + CGPA + UR + SOP + Research
- Research = GRE + TOEF + CGPA + UR + SOP + LOR

```python
display(Math(r'\begin{cases} VIF = 1 \Rightarrow No  \ existe \ una \ multicolienalidad \\ 1 < VIF < 5 \Rightarrow Existe \ una \ cierta \ multicolienalidad \
```

Python

$$\begin{cases} VIF = 1 \Rightarrow No\ existe\ una\ multicolienalidad \\ 1 < VIF < 5 \Rightarrow Existe\ una\ cierta\ multicolienalidad\ sin\ embargo\ no\ requiere\ atención \\ VIF > 5 \Rightarrow Existe\ una\ multicolienalidad \end{cases}$$

```python
slm_gre = smf.ols(formula = "GRE_Score~TOEF_Score+CGPA+SOP+LOR+University_Rating+Research", data = entrenamiento_m).fit()
r2_gre = slm_gre.rsquared
VIF_gre = 1 / (1 - r2_gre)
VIF_gre
```

Python

```
4.755208934883077
```

```python
slm_toef = smf.ols(formula = "TOEF_Score~GRE_Score+CGPA+SOP+LOR+University_Rating+Research", data = entrenamiento_m).fit()
r2_toef = slm_toef.rsquared
VIF_toef = 1 / (1 - r2_toef)
VIF_toef
```

Python

```
4.53629732956326
```

```python
slm_cgpa = smf.ols(formula = "CGPA~GRE_Score+TOEF_Score+SOP+LOR+University_Rating+Research", data = entrenamiento_m).fit()
r2_cgpa = slm_cgpa.rsquared
VIF_cgpa = 1 / (1 - r2_cgpa)
VIF_cgpa
```

Python

```
5.140960827876187
```

```python
slm_sop = smf.ols(formula = "SOP~GRE_Score+TOEF_Score+CGPA+LOR+University_Rating+Research", data = entrenamiento_m).fit()
r2_sop = slm_sop.rsquared
VIF_sop = 1 / (1 - r2_sop)
VIF_sop
```

Python

```
3.1174730861809667
```

```python
slm_lor = smf.ols(formula = "LOR~GRE_Score+TOEF_Score+CGPA+SOP+University_Rating+Research", data = entrenamiento_m).fit()
r2_lor = slm_lor.rsquared
VIF_lor = 1 / (1 - r2_lor)
VIF_lor
```

Python

```
2.484532639410723
```

```python
slm_ur = smf.ols(formula = "University_Rating~GRE_Score+TOEF_Score+CGPA+SOP+LOR+Research", data = entrenamiento_m).fit()
r2_ur = slm_ur.rsquared
VIF_ur = 1 / (1 - r2_ur)
VIF_ur
```

```
3.2614811895193743
```

```python
slm_rr = smf.ols(formula = "Research~GRE_Score+TOEF_Score+CGPA+SOP+LOR+University_Rating", data = entrenamiento_m).fit()
r2_rr = slm_rr.rsquared
VIF_rr = 1 / (1 - r2_rr)
VIF_rr
```

```
1.4973139163304312
```

Prueba de Hipotesis

```python
slm_m = smf.ols(formula = "Chance_of_Admit~GRE_Score+TOEF_Score+CGPA+SOP+LOR+University_Rating+Research", data = entrenamiento_m).fit()
slm_m.summary()
```

OLS Regression Results

Dep. Variable:	Chance_of_Admit	R-squared:	0.806
Model:	OLS	Adj. R-squared:	0.802
Method:	Least Squares	F-statistic:	178.2
Date:	Mon, 28 Nov 2022	Prob (F-statistic):	6.67e-103
Time:	16:08:54	Log-Likelihood:	413.56
No. Observations:	308	AIC:	-811.1
Df Residuals:	300	BIC:	-781.3
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.3282	0.145	-9.184	0.000	-1.613	-1.044
GRE_Score	0.0020	0.001	2.842	0.005	0.001	0.003
TOEF_Score	0.0031	0.001	2.485	0.014	0.001	0.006
CGPA	0.1167	0.014	8.605	0.000	0.090	0.143
SOP	0.0006	0.006	0.097	0.923	-0.012	0.013
LOR	0.0236	0.006	3.770	0.000	0.011	0.036
University_Rating	-0.0013	0.006	-0.231	0.818	-0.013	0.010
Research	0.0229	0.009	2.555	0.011	0.005	0.040

Omnibus:	53.988	Durbin-Watson:	1.008
Prob(Omnibus):	0.000	Jarque-Bera (JB):	90.521

Error

```python
aux2 = 0
aux2 = -1.211823 + (0.001687 * entrenamiento_m["GRE_Score"]) + (0.003719* entrenamiento_m["TOEF_Score"]) + (0.104078 * entrenamiento_m["CGPA"] ) + (    0.00509
aux2.tolist()

SSD2 = sum((entrenamiento_m["Chance_of_Admit"] - aux2)**2)

RSE2 = np.sqrt(SSD2 / (len (entrenamiento_m) - 8 ))

chance_promedio2 = np.mean(entrenamiento_m["Chance_of_Admit"])

error2 = RSE2 / chance_promedio2

error2
```

Python

```
0.08927159191549391
```

Regresión lineal modificado

Datos

```python
x_columns_mult2 = ["GRE_Score", "TOEF_Score", "CGPA", "LOR",  "Research"]

data_base = pd.read_csv(r"C:\Users\guill\Desktop\p\MC\Temas\tareas\adm_data.csv", index_col= 0)

check = (a < .8)
entrenamiento_m2 = data_base[check]
testeo_m2 = data_base[~check]

modelo_mult2 = ["Intercept" , "GRE_Score", "TOEF_Score", "CGPA",  "LOR", "Research"]

y_train_m2 = entrenamiento_m2[["Chance_of_Admit"]]
x_train_m2 = entrenamiento_m2[["GRE_Score", "TOEF_Score", "CGPA",  "LOR", "Research"]]

entrenamiento_m
```

Python

	GRE_Score	TOEF_Score	University_Rating	SOP	LOR	CGPA	Research	Chance_of_Admit
Serial No.								
1	337	118	4	4.5	4.5	9.65	1	0.92
2	324	107	4	4.0	4.5	8.87	1	0.76
3	316	104	3	3.0	3.5	8.00	1	0.72
4	322	110	3	3.5	2.5	8.67	1	0.80
5	314	103	2	2.0	3.0	8.21	0	0.65

Modelo

```python
print(list(zip(modelo_mult2, regresion_lineal(entrenamiento_m2, x_columns_mult2, y_predic))))
```

[66] Python

```
[('Intercept', -1.317408421734676), ('GRE_Score', 0.001937874575746859), ('TOEF_Score', 0.0031083570314456715), ('CGPA', 0.1162156741685143), ('LOR', 0.023435335544753855), ('Research', 0.022779590245702863)]
```

```python
slm_m2 = LinearRegression()
slm_m2.fit(x_train_m2,y_train_m2)
modelo_m2 = {}

modelo_m2 = pd.DataFrame(slm_m2.coef_ , columns = x_train_m2.columns.values.tolist())
modelo_m2["b0"] = slm_m2.intercept_

print("El modelo de regresión lineal queda de la forma:")
display(Math(r'Chance \ of \ Admit  =   -1.293335 + 0.001668     * GRE \ Score +0.00424 * TOEFL \ Score + 0.110115 * CGPA + 0.02098  LOR  + 0.030574 * Researcł

print("r^2: ",slm_m2.score(x_train_m2, y_train_m2))



modelo_m2
```

El modelo de regresión lineal queda de la forma:

$$Chance\ of\ Admit = -1.293335 + 0.001668 * GRE\ Score + 0.00424 * TOEFL\ Score + 0.110115 * CGPA + 0.02098 LOR + 0.030574 * Research$$

r^2: 0.8060991387189149

	GRE_Score	TOEF_Score	CGPA	LOR	Research	b0
0	0.001938	0.003108	0.116216	0.023435	0.02278	-1.317408

VIF

- GRE = TOEF + CGPA + UR + LOR + Research
- TOEF = GRE + CGPA + UR + LOR + Research
- CGPA = GRE + TOEF + + UR + LOR + Research
- UR = GRE + TOEF + CGPA + LOR + Research
- LOR = GRE + TOEF + CGPA + UR + Research
- Research = GRE + TOEF + CGPA + UR + LOR

```
display(Math(r'\begin{cases} VIF = 1 \Rightarrow No  \ existe \ una \ multicolienalidad \\ 1 < VIF < 5 \Rightarrow Existe \ una \ cierta \ multicolienalidad \
```

Python

Python

$$\begin{cases} VIF = 1 \Rightarrow No\, existe\, una\, multicolienalidad \\ 1 < VIF < 5 \Rightarrow Existe\, una\, cierta\, multicolienalidad\, sin\, embargo\, no\, requiere\, atención \\ VIF > 5 \Rightarrow Existe\, una\, multicolienalidad \end{cases}$$

```
slm_gre2 = smf.ols(formula = "GRE_Score~TOEF_Score+CGPA+LOR+Research", data = entrenamiento_m2).fit()
r2_gre2 = slm_gre2.rsquared
VIF_gre2 = 1 / (1 - r2_gre2)
VIF_gre2
```

Python

```
4.699834433253745
```

```
slm_toef2 = smf.ols(formula = "TOEF_Score~GRE_Score+CGPA+LOR+Research", data = entrenamiento_m2).fit()
r2_toef2 = slm_toef2.rsquared
VIF_toef2 = 1 / (1 - r2_toef2)
VIF_toef2
```

Python

```
4.269335957185872
```

```
slm_cgpa2 = smf.ols(formula = "CGPA~GRE_Score+TOEF_Score+LOR+Research", data = entrenamiento_m2).fit()
r2_cgpa2 = slm_cgpa2.rsquared
VIF_cgpa2 = 1 / (1 - r2_cgpa2)
```

```python
slm_cgpa2 = smf.ols(formula = "CGPA~GRE_Score+TOEF_Score+LOR+Research", data = entrenamiento_m2).fit()
r2_cgpa2 = slm_cgpa2.rsquared
VIF_cgpa2 = 1 / (1 - r2_cgpa2)
VIF_cgpa2
```

4.8212542309820785

```python
slm_lor2 = smf.ols(formula = "LOR~GRE_Score+TOEF_Score+CGPA+Research", data = entrenamiento_m2).fit()
r2_lor2 = slm_lor2.rsquared
VIF_lor2 = 1 / (1 - r2_lor2)
VIF_lor2
```

1.7912320694600594

```python
slm_r2 = smf.ols(formula = "Research~GRE_Score+TOEF_Score+CGPA+LOR", data = entrenamiento_m2).fit()
r2_r2 = slm_r2.rsquared
VIF_r2 = 1 / (1 - r2_r2)
VIF_r2
```

1.4903104662044104

Prueba de Hipotesis

```python
slm_m2 = smf.ols(formula = "Chance_of_Admit~GRE_Score+TOEF_Score+CGPA+LOR+Research", data = entrenamiento_m2).fit()
slm_m2.summary()
```

[74] Python

OLS Regression Results

Dep. Variable:	Chance_of_Admit	R-squared:	0.806
Model:	OLS	Adj. R-squared:	0.803
Method:	Least Squares	F-statistic:	251.1
Date:	Mon, 28 Nov 2022	Prob (F-statistic):	2.72e-105
Time:	16:08:57	Log-Likelihood:	413.53
No. Observations:	308	AIC:	-815.1
Df Residuals:	302	BIC:	-792.7
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.3174	0.133	-9.940	0.000	-1.578	-1.057
GRE_Score	0.0019	0.001	2.845	0.005	0.001	0.003
TOEF_Score	0.0031	0.001	2.548	0.011	0.001	0.006
CGPA	0.1162	0.013	8.876	0.000	0.090	0.142
LOR	0.0234	0.005	4.422	0.000	0.013	0.034

Error

```python
aux3 = 0
aux3 = -1.293335 + (0.001668 * entrenamiento_m2["GRE_Score"]) + (0.00424* entrenamiento_m2["TOEF_Score"]) + (0.110115 * entrenamiento_m2["CGPA"] ) + (0.02098 *
aux3.tolist()

SSD3 = sum((entrenamiento_m2["Chance_of_Admit"] - aux3)**2)

RSE3 = np.sqrt(SSD3 / (len (entrenamiento_m2) - 8 ))

chance_promedio3 = np.mean(entrenamiento_m2["Chance_of_Admit"])

error3 = RSE3 / chance_promedio3

error3
```

[75] Python

```
0.0888947286200136
```

Resumen

+ Code + Markdown

```python
Image(filename="resumen.png")
```

[30] ✓ 0.4s Python

Nombre	Modelo	r^2	r^2 ajustada	error %
Base	Admit = GRE + TOEF + CGPA	0.788	0.786	9.2
Completa	Admit = GRE + TOEF + CGPA + SOP + LOR + UR + Research	0.806	0.802	8.29
Modificada	Admit = GRE + TOEF + CGPA + LOR + Research	0.806	0.803	8.89

Como se puede observar en el resumen la regresion lineal modificada tiene valores de la prueba r^2 y r^2 ajustada similares al modelo mas completo asi como el porcentaje de error de datos es muy pequeño es menos del 1%, es por esto que se propone como el modelo modificado como el mejor modelo para explicar los datos