Laboratorio de Curso de Data Analysis with Python

Nombre:Francisco Javier Ramírez Arias

Matricula: A01316379

```
#Importamos los datos del modulo #02
import pandas as pd
import numpy as np
```

```
filename='/content/automobileEDA.csv'
```

```
df = pd.read_csv(filename)
df.head()
```

	symboling	normalized-losses	make	aspiration	num-of-doors	body-style	drive-wheels
0	3	122	alfa-romero	std	two	convertible	rwd
1	3	122	alfa-romero	std	two	convertible	rwd
2	1	122	alfa-romero	std	two	hatchback	rwd
3	2	164	audi	std	four	sedan	fwd
4	2	164	audi	std	four	sedan	4wd

5 rows × 29 columns

Analizando Patrones de Caracteristicas Individuales Utilizando Visualizacion

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
print(df.dtypes)
```

```
symboling              int64
normalized-losses      int64
make                  object
aspiration            object
num-of-doors          object
body-style            object
drive-wheels          object
engine-location       object
wheel-base           float64
length               float64
width                float64
height               float64
curb-weight            int64
engine-type           object
num-of-cylinders      object
engine-size            int64
fuel-system           object
bore                 float64
stroke               float64
compression-ratio    float64
horsepower           float64
peak-rpm             float64
city-mpg               int64
highway-mpg            int64
price                float64
city-L/100km         float64
horsepower-binned     object
diesel                 int64
gas                    int64
dtype: object
```

```
#Pregunta #1
df['peak-rpm'].dtypes
```

```
dtype('float64')
```

```
df.corr()
```

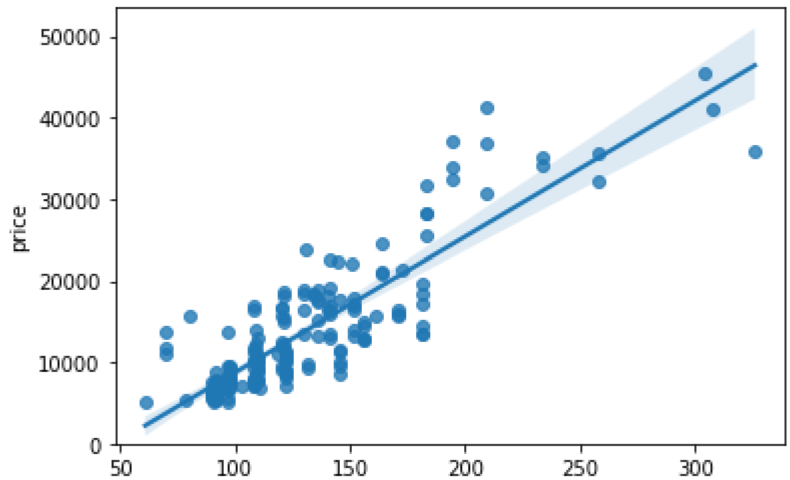
	symboling	normalized-losses	wheel-base	length	width	he
symboling	1.000000	0.466264	-0.535987	-0.365404	-0.242423	-0.55
normalized-losses	0.466264	1.000000	-0.056661	0.019424	0.086802	-0.37
wheel-base	-0.535987	-0.056661	1.000000	0.876024	0.814507	0.59
length	-0.365404	0.019424	0.876024	1.000000	0.857170	0.49
width	-0.242423	0.086802	0.814507	0.857170	1.000000	0.30
height	-0.550160	-0.373737	0.590742	0.492063	0.306002	1.00
curb-weight	-0.233118	0.099404	0.782097	0.880665	0.866201	0.30
engine-size	-0.110581	0.112360	0.572027	0.685025	0.729436	0.07
bore	-0.140019	-0.029862	0.493244	0.608971	0.544885	0.18
stroke	-0.008245	0.055563	0.158502	0.124139	0.188829	-0.06
compression-ratio	-0.182196	-0.114713	0.250313	0.159733	0.189867	0.25
horsepower	0.075819	0.217299	0.371147	0.579821	0.615077	-0.08
peak-rpm	0.279740	0.239543	-0.360305	-0.285970	-0.245800	-0.30
city-mpg	-0.035527	-0.225016	-0.470606	-0.665192	-0.633531	-0.04
highway-mpg	0.036233	-0.181877	-0.543304	-0.698142	-0.680635	-0.10
price	-0.082391	0.133999	0.584642	0.690628	0.751265	0.13
city-L/100km	0.066171	0.238567	0.476153	0.657373	0.673363	0.00
diesel	-0.196735	-0.101546	0.307237	0.211187	0.244356	0.28

```
#Pregunta #2
df[['bore', 'stroke', 'compression-ratio', 'horsepower']].corr()
```

	bore	stroke	compression-ratio	horsepower

```
#Correlacion positiva
#Tamaño del motor como variable potencial para predecir el precio
sns.regplot(x="engine-size", y="price", data=df)
plt.ylim(0,)
```

```
(0.0, 53517.59731258206)
```




```
df[["engine-size", "price"]].corr()
```

	engine-size	price
engine-size	1.000000	0.872335
price	0.872335	1.000000

```
#Millas por galon como varible predictora del precio
sns.regplot(x="highway-mpg", y="price", data=df)
```

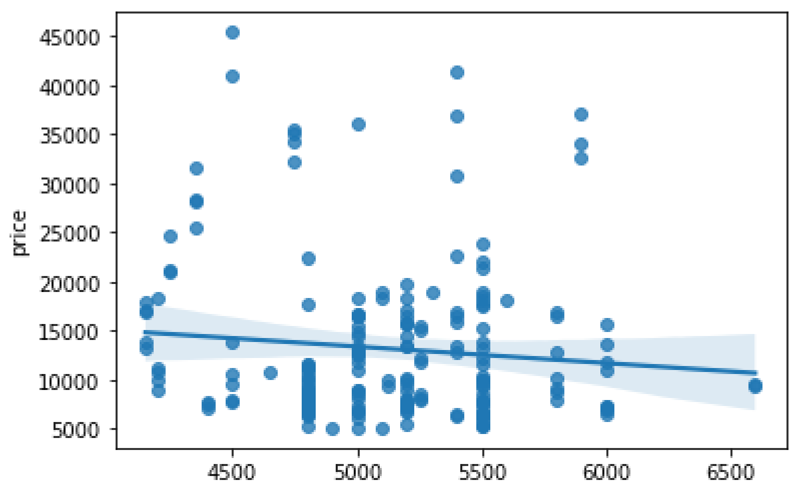
<matplotlib.axes._subplots.AxesSubplot at 0x7fe78b70bf50>


```
df[['highway-mpg', 'price']].corr()
```

	highway-mpg	price
highway-mpg	1.000000	-0.704692
price	-0.704692	1.000000

```
#Correlacion debil
sns.regplot(x="peak-rpm", y="price", data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe78b225050>




```
df[['peak-rpm','price']].corr()
```

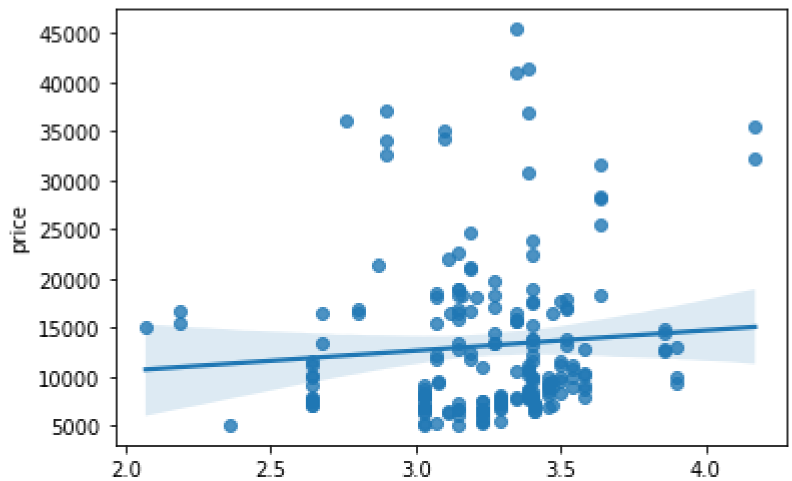
	peak-rpm	price
peak-rpm	1.000000	-0.101616
price	-0.101616	1.000000

```
#Encontrar la correlacion entre "stroke" y "price"
df[["stroke","price"]].corr()
```

	stroke	price
stroke	1.00000	0.08231
price	0.08231	1.00000

```
#Graficamos la correlacion
sns.regplot(x="stroke", y="price", data=df)
```

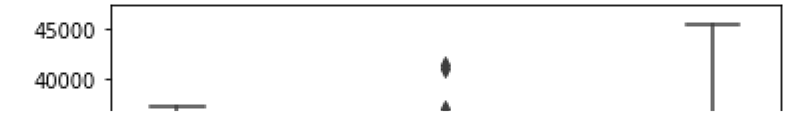
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe78b1bfd10>
```



Variables Categoricas

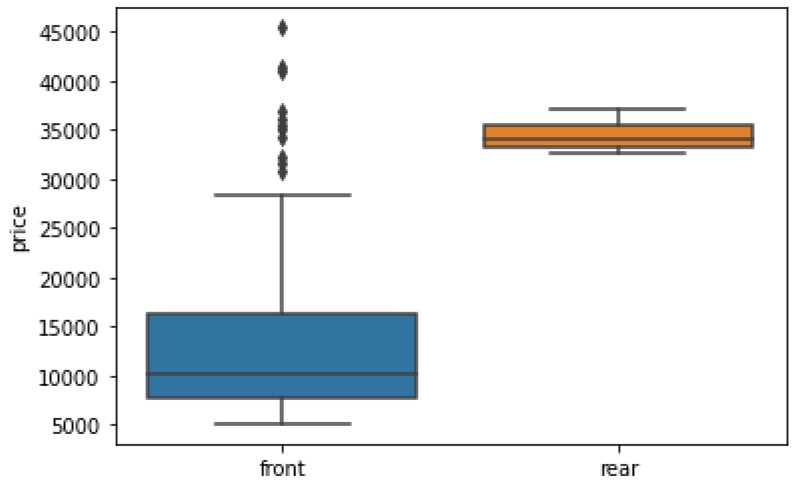
```
#Relacion entre body-style y precio
sns.boxplot(x="body-style", y="price", data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe78b12c090>



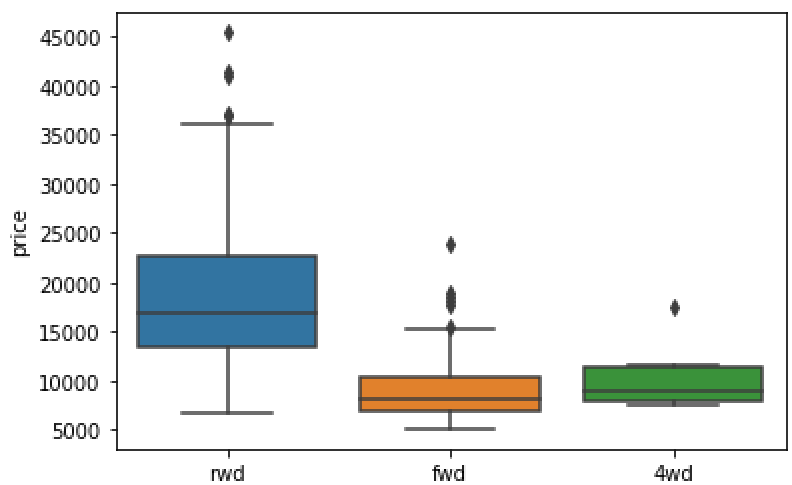

```
#RElacion entre "engine-location" y "price"
sns.boxplot(x="engine-location", y="price", data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe78b015d90>




```
#Otra relacion de variables
sns.boxplot(x="drive-wheels", y="price", data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe78afa2990>




```
#Analisis Descriptivo Estadistico
```

```
df.describe()
```

	symboling	normalized-losses	wheel-base	length	width	he
count	201.000000	201.00000	201.000000	201.000000	201.000000	201.00
mean	0.840796	122.00000	98.797015	0.837102	0.915126	53.76
std	1.254802	31.99625	6.066366	0.059213	0.029187	2.44
min	-2.000000	65.00000	86.600000	0.678039	0.837500	47.80
25%	0.000000	101.00000	94.500000	0.801538	0.890278	52.00
50%	1.000000	122.00000	97.000000	0.832292	0.909722	54.10
75%	2.000000	137.00000	102.400000	0.881788	0.925000	55.50
max	3.000000	256.00000	120.900000	1.000000	1.000000	59.80

```
df.describe(include=['object'])
```

	make	aspiration	num-of-doors	body-style	drive-wheels	engine-location	engine-type	nu cyli
count	201	201	201	201	201	201	201	
unique	22	2	2	5	3	2	6	

```
#Contamos valores
df['drive-wheels'].value_counts()
```

```
fwd    118
rwd     75
4wd      8
Name: drive-wheels, dtype: int64
```

```
df['drive-wheels'].value_counts().to_frame()
```

	drive-wheels
fwd	118
rwd	75
4wd	8

```
#Mismas acciones que los pasos anteriores, se renombrar la columna
drive_wheels_counts = df['drive-wheels'].value_counts().to_frame()
drive_wheels_counts.rename(columns={'drive-wheels': 'value_counts'}, inplace=1
drive_wheels_counts
```

	value_counts
fwd	118
rwd	75
4wd	8

```
#Renombranos el nombre de la columna de drive-wheels
drive_wheels_counts.index.name = 'drive-wheels'
drive_wheels_counts
```

	value_counts
drive-wheels	
fwd	118
rwd	75
4wd	8

```
#Los pasos anteriores pero con la columna "Engine-location"
engine_loc_counts = df['engine-location'].value_counts().to_frame()
engine_loc_counts.rename(columns={'engine-location': 'value_counts'}, inplace=
engine_loc_counts.index.name = 'engine-location'
engine_loc_counts.head(10)
```

`value_counts`

```
#Agrupamiento basico
df['drive-wheels'].unique()
```

```
array(['rwd', 'fwd', '4wd'], dtype=object)
```

```
df_group_one = df[['drive-wheels','body-style','price']]
```

```
df_group_one = df_group_one.groupby(['drive-wheels'],as_index=False).mean()
df_group_one
```

	drive-wheels	price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19757.613333

```
#agrupamos resultados
df_gptest = df[['drive-wheels','body-style','price']]
grouped_test1 = df_gptest.groupby(['drive-wheels','body-style'],as_index=False
grouped_test1
```

	drive-wheels	body-style	price
0	4wd	hatchback	7603.000000
1	4wd	sedan	12647.333333
2	4wd	wagon	9095.750000

```
#Creamos una tabla pivote
grouped_pivot = grouped_test1.pivot(index='drive-wheels',columns='body-style')
grouped_pivot
```

	price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	NaN	NaN	7603.000000	12647.333333	9095.750(

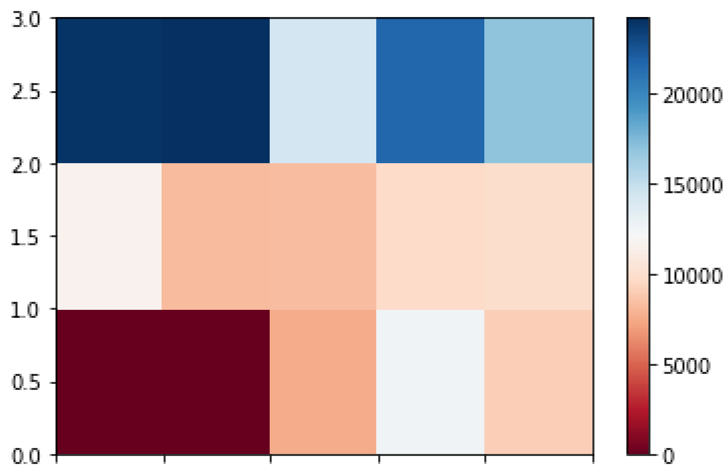
```
#Llenamos la tabla de los valores faltantes con cero
grouped_pivot = grouped_pivot.fillna(0) #fill missing values with 0
grouped_pivot
```

	price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	0.0	0.000000	7603.000000	12647.333333	9095.750(

```
#Agrupamos las variables body-stle y precio
df_gptest2 = df[['body-style','price']]
grouped_test_bodystyle = df_gptest2.groupby(['body-style'],as_index= False).me
grouped_test_bodystyle
```

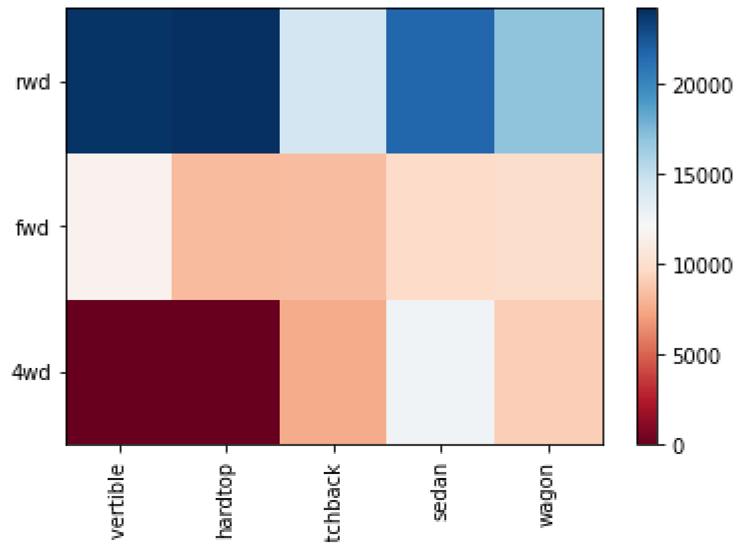
	body-style	price
0	convertible	21890.500000
1	hardtop	22208.500000
2	hatchback	9957.441176
3	sedan	14459.755319

```
import matplotlib.pyplot as plt
%matplotlib inline


#Heatmap
plt.pcolor(grouped_pivot, cmap='RdBu')
plt.colorbar()
plt.show()
```




```
fig, ax = plt.subplots()
im = ax.pcolor(grouped_pivot, cmap='RdBu')

#label names
row_labels = grouped_pivot.columns.levels[1]
col_labels = grouped_pivot.index

#move ticks and labels to the center
ax.set_xticks(np.arange(grouped_pivot.shape[1]) + 0.5, minor=False)
ax.set_yticks(np.arange(grouped_pivot.shape[0]) + 0.5, minor=False)

#insert labels
```

```
ax.set_xticklabels(row_labels, minor=False)
ax.set_yticklabels(col_labels, minor=False)

#rotate label if too long
plt.xticks(rotation=90)

fig.colorbar(im)
plt.show()
```




```
#Correlacion
df.corr()
```

	symboling	normalized-losses	wheel-base	length	width	he
symboling	1.000000	0.466264	-0.535987	-0.365404	-0.242423	-0.55
normalized-losses	0.466264	1.000000	-0.056661	0.019424	0.086802	-0.37
wheel-base	-0.535987	-0.056661	1.000000	0.876024	0.814507	0.59
length	-0.365404	0.019424	0.876024	1.000000	0.857170	0.49
width	-0.242423	0.086802	0.814507	0.857170	1.000000	0.30
height	-0.550160	-0.373737	0.590742	0.492063	0.306002	1.00
curb-weight	-0.233118	0.099404	0.782097	0.880665	0.866201	0.30
engine-size	-0.110581	0.112360	0.572027	0.685025	0.729436	0.07
bore	-0.140019	-0.029862	0.493244	0.608971	0.544885	0.18
stroke	-0.008245	0.055563	0.158502	0.124139	0.188829	-0.06
compression-ratio	-0.182196	-0.114713	0.250313	0.159733	0.189867	0.25

```
from scipy import stats
```

```
#Calculamos el coeficiente de pearson
pearson_coef, p_value = stats.pearsonr(df['wheel-base'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value
```

```
The Pearson Correlation Coefficient is 0.584641822265508  with a P-value
```

```
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value
```

```
The Pearson Correlation Coefficient is 0.8095745670036559  with a P-valu
```

```
pearson_coef, p_value = stats.pearsonr(df['length'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value
```

```
The Pearson Correlation Coefficient is 0.6906283804483638  with a P-valu
```

```
pearson_coef, p_value = stats.pearsonr(df['width'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value
```

```
The Pearson Correlation Coefficient is 0.7512653440522673  with a P-valu
```

```
pearson_coef, p_value = stats.pearsonr(df['curb-weight'], df['price'])
print( "The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is 0.8344145257702843  with a P-valu
```

```
pearson_coef, p_value = stats.pearsonr(df['engine-size'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value
```

```
The Pearson Correlation Coefficient is 0.8723351674455185  with a P-valu
```

```
pearson_coef, p_value = stats.pearsonr(df['bore'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value
```

```
The Pearson Correlation Coefficient is 0.5431553832626602  with a P-valu
```

```
pearson_coef, p_value = stats.pearsonr(df['city-mpg'], df['price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value
```

```
The Pearson Correlation Coefficient is -0.6865710067844678  with a P-val
```

```
pearson_coef, p_value = stats.pearsonr(df['highway-mpg'], df['price'])
print( "The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is -0.704692265058953  with a P-valu
```

```
#Analisis ANOVA
grouped_test2=df_gptest[['drive-wheels', 'price']].groupby(['drive-wheels'])
grouped_test2.head(2)
```

	drive-wheels	price
0	rwd	13495.0
1	rwd	16500.0
3	fwd	13950.0
4	4wd	17450.0
5	fwd	15250.0
136	4wd	7603.0

```
df_gptest
```

	drive-wheels	body-style	price
0	rwd	convertible	13495.0
1	rwd	convertible	16500.0
2	rwd	hatchback	16500.0
3	fwd	sedan	13950.0
4	4wd	sedan	17450.0
...	...	...	...
196	rwd	sedan	16845.0
197	rwd	sedan	19045.0
198	rwd	sedan	21485.0
199	rwd	sedan	22470.0
200	rwd	sedan	22625.0

```
grouped_test2.get_group('4wd')['price']
```

```
4      17450.0
136     7603.0
140     9233.0
141    11259.0
144     8013.0
145    11694.0
150     7898.0
```

```
151     8778.0
Name: price, dtype: float64
```

```python
f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'], groupec

print( "ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 67.95406500780399 , P = 3.3945443577151245e-23
```

```python
f_val, p_val = stats.f_oneway(grouped_test2.get_group('fwd')['price'], groupec

print( "ANOVA results: F=", f_val, ", P =", p_val )
```

```
ANOVA results: F= 130.5533160959111 , P = 2.2355306355677845e-23
```

```python
f_val, p_val = stats.f_oneway(grouped_test2.get_group('4wd')['price'], groupec

print( "ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 8.580681368924756 , P = 0.004411492211225333
```

```python
f_val, p_val = stats.f_oneway(grouped_test2.get_group('4wd')['price'], groupec

print("ANOVA results: F=", f_val, ", P =", p_val)
```

```
ANOVA results: F= 0.665465750252303 , P = 0.41620116697845666
```

Exploratory Data Analysis (EDA) Module 3

Learning objectives

- Descriptive Statistics
  * Group by
  * ANOVA
- Correlation
  * Correlation - Statistics

Descriptive Statistics

* Describe basic features of data
* Give short summaries about the sample ~~&~~ and measures of the data.

Describe() → function

* Summarize statistics using pandas describe() method

df.describe()

Value Counts()

* Summarize the categorical data is by using the value_counts() method

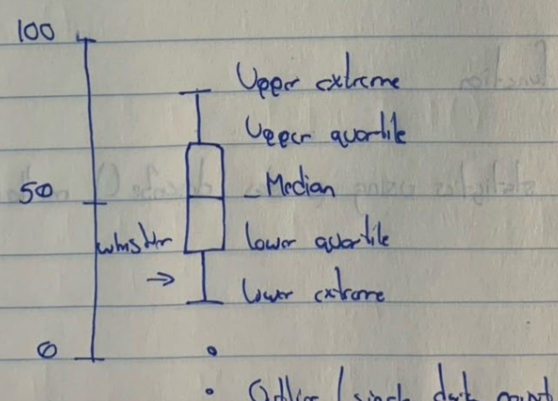
```
drive_wheels_counts = df["drive-wheels"].value_counts()

drive_wheels_counts.rename(columns={'drive-wheels':'value_counts' inplace=True)
drive_wheels_counts.index.name = 'drive-wheels'
```

drive-wheels	value_counts
fwd	118
rwd	75
4wd	8

Descriptive-Statistics

Box-Plots

Scale



```
sns.boxplot(x = "drive-wheels", y = "price", data = df)
```

Scatter - plot

* Each observation represented as a point
* Scatter plot show the relationship between two variables.

1. Predictor / Independent variables on x-axis.
2. Target / dependent variables on y-axis.

```
y = df["engine-size"]
x = df["price"]
plt.scatter(x, y)

plt.title("Scatterplot of Engine Size vs Price")
plt.xlabel("Engine Size")
plt.ylabel("Price")
```

Grouping data

→ Use Panda dataframe. Groupby() method:

· Can be applied on categorical variables
· Group data into categories
· Single or multiple variables

Groupby()

```
df_test = df[['drive-wheels', 'body-style', 'price']]
df_grp = df_test.groupby(['drive-wheels', 'body-style'], as_index=False)
         .mean()

df_grp
```

Pandas method - Pivot()

- One variable displayed along the columns and the other variable displayed along the rows

```
df_pivot = df_grp.pivot(index='drive-wheels', columns='body-style')
```

Heatmap

Plot target variable over multiple variables

```
plt.pcolor(df_pivot cmap = 'RdBu")
plt.colorbar()
plt.show()
```

ANOVA

Analysis of Variance (Anova)

- Statistical comparison of groups

- Analysis of Variance (ANOVA)

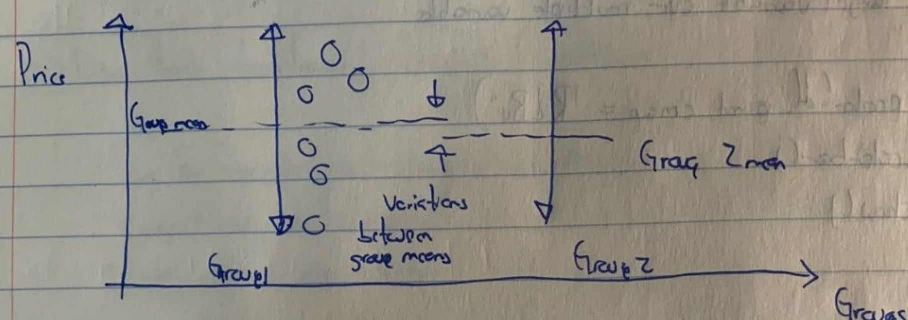
* Why do we perform ANOVA?
  - Finding correlation between different groups of categorical variables

* What we obtain from Anova?

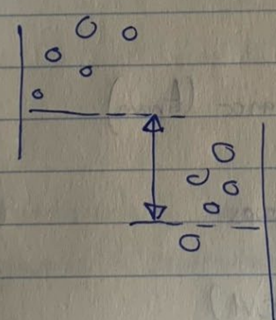
  - F-test score: variation between sample group means divided by variation within sample group.
  - p-value: confidence degree.

F-test

- Small F imply poor correlation between variable categories and target variable



Large F imply strong correlation between variable categories and target variable



ANOVA

- Anova between "Honda" and "Subaru"

```
df_anova = df[["make", "price"]]
grouped_anova = df_anova.groupb([["make"]])

anova_results_1 = stats.f_oneway(grouped_anova.get_group("honda")["price"], grouped_anova.get_group
    ("subaru")["price"])
```

Correlation

What is correlation?

- Measure to what extend different variables are interdependent.

- For example

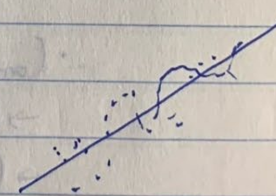
  → Lung cancer → Smoking

  Rain → Umbrella

- Correlation doesn't imply causation.

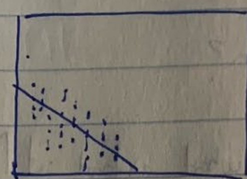
Correlation - Positive Linear Relationship
- Correlation between two features.

```
sns.regplot(x="engine-size", y="prices", data=df)
plt.ylim(Ø,)
```



Correlation - Negative Linear Relationship

```
sns.regplot(x= "highway-mpg", y= "price", data=df)
plt.ylim(0,)
```



*Weak correlation between two features

```
sns.regplot(x="peak-rpm", y= "price", data=df)
plt.ylim(0,)
```

Correlation - Statistics

Pearson Correlation

- Measure the strength of the correlation between two features.
  → Correlation coefficient
  → P-value

- Correlation coefficient
  → Close to +1: Large positive relationship
  → Close to -1: Large negative relationship
  - Close to 0: No relationship.

· P-value

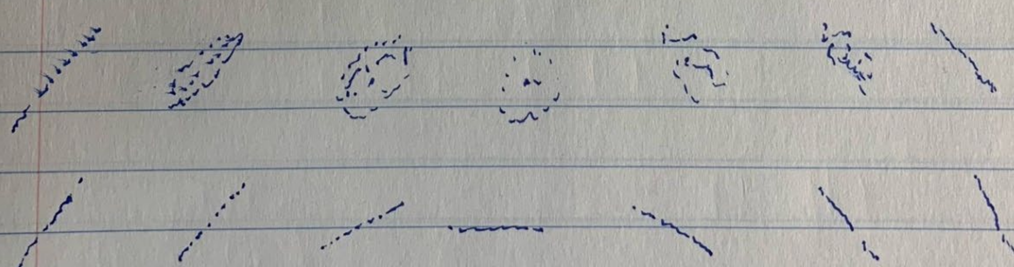
· P-value < 0.001 Strong certainty in the result
· P-value < 0.05 Moderate certainty in the result
· P-value < 0.1 Weak certainty in the result
· P-value > 0.1 No certainty in the result

Strong Correlation

- Correlation Coefficient close to 1 or -1
- P-value less than 0.001

Pearson Correlation

1 0.8 0.4 0 -0.4 -0.8 -1



1 1 1 -1 -1 -1

Pearson Correlation

```
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
```

- Pearson correlation: 0.81
- P-value: 9.35e-48

Correlation Heatmap →

Course Progress Dates Discussion

Course Progress for 'Francisco_Arias' (A01316379@tec.mx)