😎 Análisis de datos y distribución de variables

Los datos para este ejemplo se pueden obtener aquí (https://www.fueleconomy.gov/feg/ws/index.shtml)

Paso 1: cargamos los datos, renombramos las columnas, obtenemos el tamaño del dataset y exportamos un nuevo archivo.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

vehiculos = pd.read_csv("archs/vehiculos_original.csv")
vehiculos.head()
```

Out[1]:

	make	model	year	displ	cylinders	trany	drive	VClass	fuelType	comb08	co2TailpipeGpm
0	AM General	DJ Po Vehicle 2WD	1984	2.5	4.0	Automatic 3-spd	2-Wheel Drive	Special Purpose Vehicle 2WD	Regular	17	522.764706
1	AM General	DJ Po Vehicle 2WD	1984	2.5	4.0	Automatic 3-spd	2-Wheel Drive	Special Purpose Vehicle 2WD	Regular	17	522.764706
2	AM General	FJ8c Post Office	1984	4.2	6.0	Automatic 3-spd	2-Wheel Drive	Special Purpose Vehicle 2WD	Regular	13	683.615385
3	AM General	FJ8c Post Office	1984	4.2	6.0	Automatic 3-spd	2-Wheel Drive	Special Purpose Vehicle 2WD	Regular	13	683.615385
4	AM General	Post Office DJ5 2WD	1985	2.5	4.0	Automatic 3-spd	Rear-Wheel Drive	Special Purpose Vehicle 2WD	Regular	16	555.437500

In [2]:
```
vehiculos = vehiculos.rename(columns={
    "cylinders": "cilindros",
    "trany": "transmision",
    "make": "fabricante",
    "model": "modelo",
    "displ": "desplazamiento",
    "drive": "traccion",
    "VClass": "clase",
    "fuelType": "combustible",
    "comb08": "consumo",
    "co2TailpipeGpm": "co2"
    })

vehiculos.to_csv("archs/vehiculos_1.csv", index=False)
```

In [3]:
```
vehiculos.shape
```

Out[3]: (38436, 11)

Paso 2: revisamos la calidad de los datos (QA), que no existan datos duplicados (no deberían existir), datos nulos o inexistentes (que pudiesen arrojarnos errores de procesamiento) y se revisan también los valores extremos.

In [4]:
```
vehiculos[vehiculos.duplicated()].shape
```

Out[4]: (1506, 11)

Eliminamos duplicados:

In [5]:
```
vehiculos = vehiculos.drop_duplicates()
vehiculos.shape
```

Out[5]: (36930, 11)

Medimos la **cardinalidad** de algunas columnas, con esto queremos observar si una cantidad grande de registros tiene el mismo valor:

In [6]:
```
print("Revisando valores duplicados:\n")
n_registros = len(vehiculos)

def valores_duplicados_col(df):
    for columna in df:
        n_por_valor = df[columna].value_counts()
        mas_comun = n_por_valor.iloc[0]
        menos_comun = n_por_valor.iloc[-1]
        print ("{} | {} - {} | {}".format(
            df[columna].name,
            round(mas_comun/(1.0 * n_registros),3),
            round(menos_comun/(1.0 * n_registros),3),
            df[columna].dtype))

valores_duplicados_col(vehiculos)
```

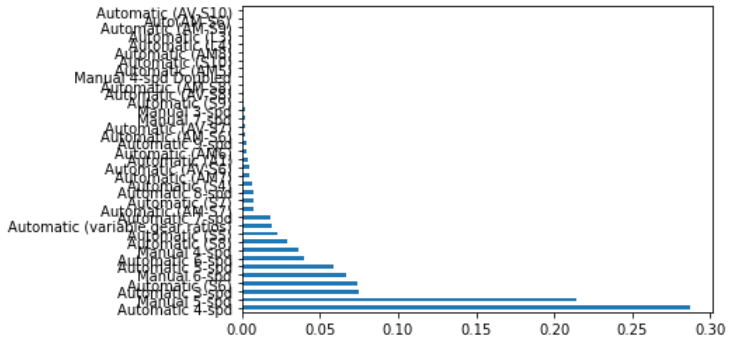
```
Revisando valores duplicados:

fabricante | 0.1 - 0.0 | object
modelo | 0.005 - 0.0 | object
year | 0.038 - 0.007 | int64
desplazamiento | 0.095 - 0.0 | float64
cilindros | 0.38 - 0.0 | float64
transmision | 0.287 - 0.0 | object
traccion | 0.353 - 0.005 | object
clase | 0.145 - 0.0 | object
combustible | 0.652 - 0.0 | object
consumo | 0.097 - 0.0 | int64
co2 | 0.084 - 0.0 | float64
```

Podemos graficar columna por columna, por ejemplo graficamos las columnas "transmision", "cilindros" y "combustible":

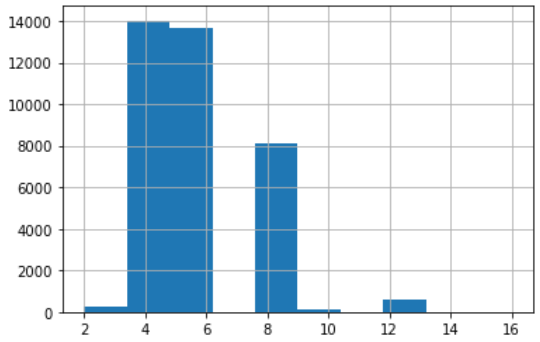
In [7]:
```
vehiculos.transmision.value_counts(normalize=True).plot.barh()
```

Out[7]: <AxesSubplot:>



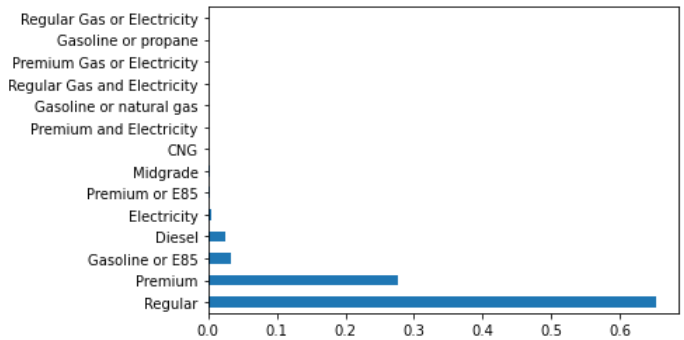
In [8]:
```
vehiculos.cilindros.hist()
```

Out[8]: <AxesSubplot:>



In [9]:
```
vehiculos.combustible.value_counts(normalize=True).plot.barh()
```

Out[9]: <AxesSubplot:>



Revisamos el porcentaje de valores inexistentes en cada columna:

In [10]:
```
print ("Revisando porcentajes de valores inexistentes:\n")
n_registros = len(vehiculos)

def valores_inexistentes_col(df):
    for columna in df:
        print ("{} | {} |{}".format(
            df[columna].name,
            len(df[df[columna].isnull()]) / (1.0 * n_registros),
            df[columna].dtype))

valores_inexistentes_col(vehiculos)
```

```
Revisando porcentajes de valores inexistentes:

fabricante | 0.0 |object
modelo | 0.0 |object
year | 0.0 |int64
desplazamiento | 0.0037909558624424585 |float64
cilindros | 0.003845112374763065 |float64
transmision | 0.00029786081776333605 |object
traccion | 0.02158137015976171 |object
clase | 0.0 |object
combustible | 0.0 |object
consumo | 0.0 |int64
co2 | 0.0 |float64
```

Se consideran **valores extremos** a aquellos cuya puntuación Z es mayor a 3. Se considera a Z como:

Z = (x – media) / desviación_estándar

In [11]:
```
print ("Revisando los valores extremos:\n")

def extermos_col(df):
    for columna in df:
        if df[columna].dtype != object:
            n_extremos = len(df[np.abs(stats.zscore(df[columna]))>3])
            print ("{} | {} | {}".format(
                df[columna].name,
                n_extremos,
                df[columna].dtype))

extermos_col(vehiculos)
```

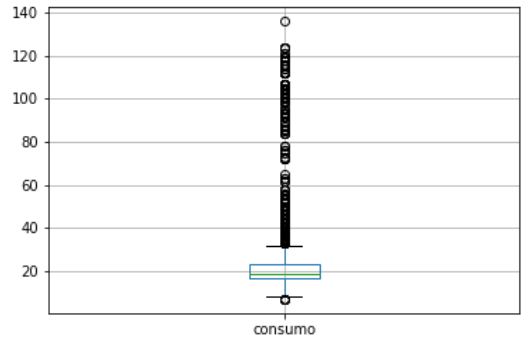
```
Revisando los valores extremos:

year | 0 | int64
desplazamiento | 0 | float64
cilindros | 0 | float64
consumo | 233 | int64
co2 | 358 | float64
```

Graficamos:

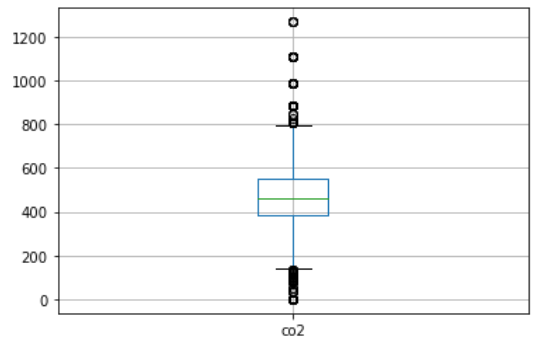
In [12]:
```
vehiculos.boxplot(column='consumo')
```

Out[12]: <AxesSubplot:>



In [13]:
```
vehiculos.boxplot(column='co2')
```

Out[13]: <AxesSubplot:>



¿Qué combustible usarán los vehículos que no producen co2?:

In [14]:
```
vehiculos[vehiculos.co2==0].combustible.unique()
```

Out[14]:
```
array(['Electricity'], dtype=object)
```

Los tipos de combustible son:

In [15]:
```
vehiculos.combustible.unique()
```

Out[15]:
```
array(['Regular', 'Premium', 'Diesel', 'Premium and Electricity',
       'Premium or E85', 'Electricity', 'Premium Gas or Electricity',
       'Gasoline or E85', 'Gasoline or natural gas', 'CNG',
       'Regular Gas or Electricity', 'Midgrade',
       'Regular Gas and Electricity', 'Gasoline or propane'], dtype=object)
```

Nos interesan los vehículos que sí generan co2, por lo tanto vamos a eliminar a los que no generan co2:

In [16]:
```
vehiculos[vehiculos.co2==0].shape
```

Out[16]:
```
(139, 11)
```

In [17]:
```
vehiculos_no_electricos = vehiculos[vehiculos.co2>0]
vehiculos_no_electricos.shape
```

Out[17]:
```
(36791, 11)
```

Hasta aquí:

- Hay 1506 registros duplicados que se han eliminado.
- Las variables desplazamiento, cilindro, transmision y traccion tienen valores nulos o inexistentes.
- La variable combustible tiene una clase dominante en 65%.
- Se eliminaron 139 registros correspondientes a autos electricos que no emiten co2.

In [18]:
```
vehiculos_no_electricos.to_csv("archs/vehiculos_2.csv", index=False)
```

Paso 3: Se requiere **agrupar algunas variables** a fin de reducir el campo de estudio a aquellas variables que nos interesan:

In [19]:
```
print ("Visualizar los valores únicos por columna:\n")

def valores_unicos_col(df):
    for column in df:
        print ("{} | {} | {} ".format(
            df[column].name, len(df[column].unique()), df[column].dtype))

valores_unicos_col(vehiculos)
```

```
Visualizar los valores únicos por columna:

fabricante | 133 | object
modelo | 3791 | object
year | 35 | int64
desplazamiento | 67 | float64
cilindros | 10 | float64
transmision | 38 | object
traccion | 8 | object
clase | 34 | object
combustible | 14 | object
consumo | 84 | int64
co2 | 597 | float64
```

Agrupamiento de variables categóricas

Primer caso: la columna "clase:

In [20]:
```
print ("Visualizar los valores de la columna clase:\n")
vehiculos.clase.unique()
```

```
Visualizar los valores de la columna clase:
```

Out[20]:
```
array(['Special Purpose Vehicle 2WD', 'Midsize Cars', 'Subcompact Cars',
       'Compact Cars', 'Sport Utility Vehicle - 4WD',
       'Small Sport Utility Vehicle 2WD',
       'Small Sport Utility Vehicle 4WD', 'Two Seaters',
       'Sport Utility Vehicle - 2WD', 'Special Purpose Vehicles',
       'Special Purpose Vehicle 4WD', 'Small Station Wagons',
       'Minicompact Cars', 'Midsize-Large Station Wagons',
       'Midsize Station Wagons', 'Large Cars',
       'Standard Sport Utility Vehicle 4WD',
       'Standard Sport Utility Vehicle 2WD', 'Minivan - 4WD',
       'Minivan - 2WD', 'Vans', 'Vans, Cargo Type',
       'Vans, Passenger Type', 'Standard Pickup Trucks 2WD',
       'Standard Pickup Trucks', 'Standard Pickup Trucks/2wd',
       'Small Pickup Trucks 2WD', 'Standard Pickup Trucks 4WD',
       'Small Pickup Trucks 4WD', 'Small Pickup Trucks', 'Vans Passenger',
       'Special Purpose Vehicle', 'Special Purpose Vehicles/2wd',
       'Special Purpose Vehicles/4wd'], dtype=object)
```

Se puede clasificar todos éstos valores en ciertas categorías. Crearemos las siguientes categorías o tipos para la columna clase:

- Coches pequeños
- Coches medianos
- Coches grandes
- Camionetas
- Vehículos especiales
- Deportivos
- Coche familiar
- Furgoneta

A continuación creamos una nueva columna llamada "clase_tipo", donde se clasificará cada registro en su tipo correspondiente:

In [21]:
```
pequeno = ['Compact Cars', 'Subcompact Cars', 'Two Seaters', 'Minicompact Cars']
medio = ['Midsize Cars']
grande= ['Large Cars']

vehiculos.loc[vehiculos['clase'].isin(pequeno),'clase_tipo'] = 'Coches pequeños'
vehiculos.loc[vehiculos['clase'].isin(medio),'clase_tipo'] = 'Coches medianos'
vehiculos.loc[vehiculos['clase'].isin(grande),'clase_tipo'] = 'Coches grandes'

vehiculos.loc[vehiculos['clase'].str.contains('Truck'), 'clase_tipo'] = 'Camionetas'
vehiculos.loc[vehiculos['clase'].str.contains('Special Purpose'), 'clase_tipo'] = 'Vehiculos especiales'
vehiculos.loc[vehiculos['clase'].str.contains('Sport Utility'), 'clase_tipo'] = 'Deportivos'
vehiculos.loc[vehiculos['clase'].str.contains('Station'), 'clase_tipo'] = 'Coche familiar'
vehiculos.loc[vehiculos['clase'].str.lower().str.contains('van'), 'clase_tipo'] = 'Furgoneta'
```

Es útil que las columnas categóricas sean consideradas como tales por Python. A continuación haremos eso a la columna “clase_tipo”:

In [22]:
```
vehiculos.clase_tipo = vehiculos.clase_tipo.astype("category")
```

Si deseamos verificar que todas las filas hayan recibido una categoría, podemos contar cuántos registros contiene cada una de ellas.

In [23]:
```
print ("Mis categorías de clase:\n")
vehiculos.clase_tipo.value_counts()
```

```
Mis categorías de clase:
```

Out[23]:
```
Coches pequeños         13055
Camionetas               5446
Deportivos               5313
Coches medianos          4274
Coche familiar           2540
Vehiculos especiales     2216
Furgoneta                2213
Coches grandes           1873
Name: clase_tipo, dtype: int64
```

Segundo caso: la columna "traccion", agrupamos los registros en dos tipos: dos y cuatro ruedas.

In [24]:
```
print ("Valores de la columna traccion:\n")
vehiculos.traccion.unique()
```

```
Valores de la columna traccion:
```

Out[24]:
```
array(['2-Wheel Drive', 'Rear-Wheel Drive', 'Front-Wheel Drive',
       '4-Wheel or All-Wheel Drive', 'All-Wheel Drive', nan,
       '4-Wheel Drive', 'Part-time 4-Wheel Drive'], dtype=object)
```

In [25]:
```
print ("Mis Categorias de tracción:\n")

vehiculos["traccion_tipo"] = "dos"
vehiculos.loc[vehiculos.traccion.isin([
    "4-Wheel or All-Whell Drive", "All-Whell Drive",
    "4-Wheel Drive", "Part Time 4-Whell Drive"]), "traccion_tipo"] = "cuatro"

vehiculos.traccion_tipo.value_counts()
```

```
Mis Categorias de tracción:


dos       35814
cuatro     1116
Name: traccion_tipo, dtype: int64
```

Tercer caso: la columna "transmision", necesitamos solo dos categorías (Manual y Automática):

In [26]:
```
print ("Mis Categorias de transmisión:\n")
vehiculos["transmision_tipo"] = "Automática"
vehiculos.loc[vehiculos.transmision.notnull() & vehiculos.transmision.str.startswith('M'), "transmision_tipo"] = "Manual"
vehiculos.transmision_tipo.value_counts()
```

```
Mis Categorias de transmisión:


Automática    25076
Manual        11854
Name: transmision_tipo, dtype: int64
```

Cuarto caso: para la columna "combustible" creamos los tipos: Normal, Premium, Híbrido y Otros tipos de combustible:

In [27]:
```
print ("Mis Categorias de combustible:\n")
vehiculos["combustible_tipo"] = "Otros tipos de combustible"
vehiculos.loc[vehiculos["combustible"] == "Regular", "combustible_tipo"] = "Normal"
vehiculos.loc[vehiculos["combustible"] == "Premium", "combustible_tipo"] = "Premium"
vehiculos.loc[vehiculos["combustible"].str.contains("Electricity"), "combustible_tipo"]= "Hibrido"
vehiculos.combustible_tipo.value_counts()
```

```
Mis Categorias de combustible:


Normal                        24078
Premium                       10206
Otros tipos de combustible     2437
Hibrido                         209
Name: combustible_tipo, dtype: int64
```

Agrupamiento de variables continuas

Las variables continuas tienen valores numéricos por lo que se deben clasificar de acuerdo a rangos de valores. Esto ocurre en el caso de las columnas "desplazamiento", "consumo" y "co2". Éstas columnas las clasificaremos en categorías como:

- muy bajo
- bajo
- moderado
- alto
- muy alto

In [28]:
```
print ("Mis categorías de desplazamiento:\n")
tipos_tam_motor = ["muy pequeño", "pequeño", "mediano", "grande", "muy grande"]
vehiculos["tamano_motor_tipo"] = pd.qcut(vehiculos["desplazamiento"], 5, tipos_tam_motor)
vehiculos.tamano_motor_tipo.value_counts()
```

```
Mis categorías de desplazamiento:


muy pequeño    8246
mediano        7892
muy grande     7002
grande         6997
pequeño        6653
Name: tamano_motor_tipo, dtype: int64
```

In [29]:
```
print ("Mis categorías de consumo:\n")
tipos_consumo = ["muy bajo", "bajo", "moderado","alto", "muy alto"]
vehiculos["consumo_tipo"] = pd.qcut(vehiculos["consumo"], 5, tipos_consumo)
vehiculos.consumo_tipo.value_counts()
```

```
Mis categorías de consumo:


moderado    9760
muy bajo    9151
muy alto    6152
bajo        6003
alto        5864
Name: consumo_tipo, dtype: int64
```

In [30]:
```
print ("Mis categorías de co2:\n")
tipos_co2 = ["muy bajo", "bajo", "moderado","alto", "muy alto"]
vehiculos["co2_tipo"] = pd.qcut(vehiculos["co2"], 5, tipos_co2)
vehiculos.co2_tipo.value_counts()
```

```
Mis categorías de co2:
```

Out[30]:
```
moderado    10118
muy bajo     7461
bajo         7437
muy alto     7152
alto         4762
Name: co2_tipo, dtype: int64
```

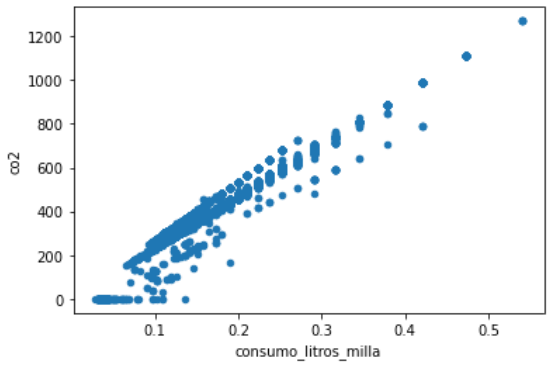
Por último ajustaremos los valores de consumo y co2 para que ambos utilicen una misma unidad de medida:

In [31]:
```
litros_por_galon = 3.78541
vehiculos["consumo_litros_milla"] = litros_por_galon / vehiculos.consumo
vehiculos.head()
```

Out[31]:

	fabricante	modelo	year	desplazamiento	cilindros	transmision	traccion	clase	combustible	consumo	co2	clase_tipo	traccion_tipo	transm
0	AM General	DJ Po Vehicle 2WD	1984	2.5	4.0	Automatic 3-spd	2-Wheel Drive	Special Purpose Vehicle 2WD	Regular	17	522.764706	Vehiculos especiales	dos	
2	AM General	FJ8c Post Office	1984	4.2	6.0	Automatic 3-spd	2-Wheel Drive	Special Purpose Vehicle 2WD	Regular	13	683.615385	Vehiculos especiales	dos	
4	AM General	Post Office DJ5 2WD	1985	2.5	4.0	Automatic 3-spd	Rear-Wheel Drive	Special Purpose Vehicle 2WD	Regular	16	555.437500	Vehiculos especiales	dos	
5	AM General	Post Office DJ8 2WD	1985	4.2	6.0	Automatic 3-spd	Rear-Wheel Drive	Special Purpose Vehicle 2WD	Regular	13	683.615385	Vehiculos especiales	dos	
6	ASC Incorporated	GNX	1987	3.8	6.0	Automatic 4-spd	Rear-Wheel Drive	Midsize Cars	Premium	16	555.437500	Coches medianos	dos	

In [32]:
```
vehiculos.plot.scatter(x="consumo_litros_milla", y = "co2")
plt.show()
```



Guardamos los datos en formato pickle para evitar que se pierda información sobre qué variables son categóricas

In [33]:
```
vehiculos.to_pickle("archs/vehiculos_3.pkl")
```

Paso 4: Distribución de variables. Verificamos como lee pandas los datos, es decir, qué tipo de dato le asigna a cada uno.

In [34]:
```
vehiculos = pd.read_pickle("archs/vehiculos_3.pkl")
vehiculos.dtypes
```

Out[34]:
```
fabricante              object
modelo                  object
year                     int64
desplazamiento         float64
cilindros              float64
transmision             object
traccion                object
clase                   object
combustible             object
consumo                  int64
co2                    float64
clase_tipo            category
traccion_tipo           object
transmision_tipo        object
combustible_tipo        object
tamano_motor_tipo     category
consumo_tipo          category
co2_tipo              category
consumo_litros_milla   float64
dtype: object
```

Observamos que hay variables numéricas y variables categóricas (que creamos en el paso anterior). Lo que vamos a hacer es analizar la distribución de las variables

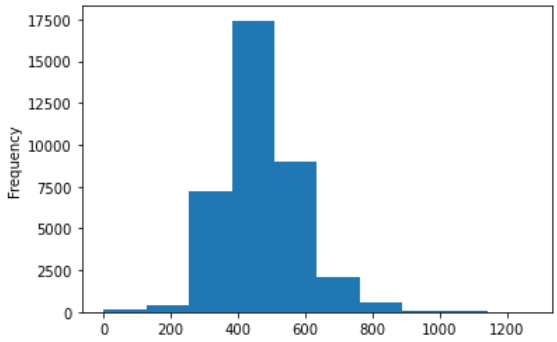
Distribución de variables numéricas

Histogramas

Podemos crear un histograma para ver la distribución de una variable. Por ejemplo, tenemos una columna llamada co2, donde se registra el nivel de contaminación que emite cada vehículo. ¿Cómo es la distribución de ésta variable? Creamos un histograma para esa columna:

In [35]:
```
vehiculos['co2'].plot.hist()
```

Out[35]:
```
<AxesSubplot:ylabel='Frequency'>
```



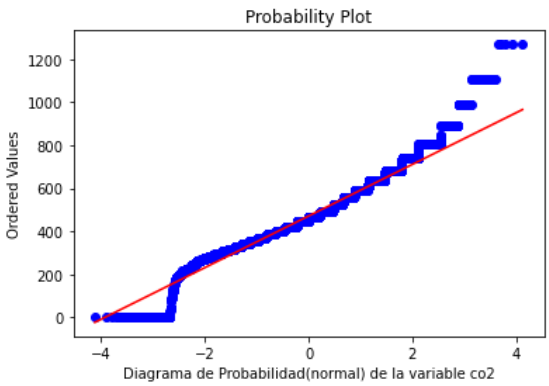
Podemos hacer lo mismo con el resto de variables numéricas. Observamos que al parecer siguen una distribución normal ya que tiene la típica forma de "campana" que tiene una distribución normal. Es importante saber si nuestras variables siguen una ésta distribución dado que muchos algoritmos la asumen.

Gráfico de probabilidad

Otra forma de comprobar esto es usando un gráfico de probabilidad. Probamos con la columna "co2". Cuanto más se parezca nuestra gráfica a una línea de 45 grados, más normal será:

In [36]:
```
def normalidad_variable_numerica(col):
    stats.probplot(vehiculos[col], plot=plt)
    plt.xlabel('Diagrama de Probabilidad(normal) de la variable {}'.format(col))
    plt.show()

normalidad_variable_numerica('co2')
```



En el caso de la variable co2, se ajusta bastante a una distribución normal.

Test de normalidad

Otra forma de asegurarse de que se sigue una distribución normal es realizando un test de normalidad:

In [37]:
```
columnas_numericas = vehiculos.select_dtypes(['int', 'float']).columns
for num_col in columnas_numericas:
    _, pval = stats.normaltest(vehiculos[num_col])
    if(pval < 0.05):
        print("Columna {} no sigue una distribución normal".format(num_col))
```

```
Columna year no sigue una distribución normal
Columna consumo no sigue una distribución normal
Columna co2 no sigue una distribución normal
Columna consumo_litros_milla no sigue una distribución normal
```

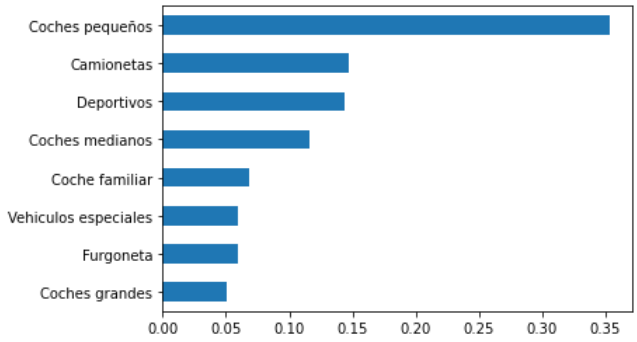
En un test de normalidad, aceptamos o rechazamos que una variable sigue una distribución normal, para, en el caso de un nivel de confianza de 95%, el pval sea menor o mayor a 0.05. El código anterior está diseñado para probar todas las columnas numéricas que encuentre. Como se puede observar, estrictamente hablando, ninguna de nuestras variables sigue una distribución normal.

Distribución de variables categóricas

La siguiente función nos permite observar la distribución de una variable categórica. Probemos con clase_tipo, una variable donde clasificamos los tipos de vehículos disponibles:

In [38]:
```
def distribucion_variable_categorica(col):
    vehiculos[col].value_counts(ascending=True,normalize=True).tail(20).plot.barh()
    plt.show()

distribucion_variable_categorica('clase_tipo')
```



En éste caso, observamos que la clase mayoritaria de vehículos es la de coches pequeños, con un 35% del total.

Otros ejemplos (https://www.cienciadedatos.net/documentos/pystats01-ajuste-distribuciones-python.html)