```
import pandas as pd
import numpy as np

# Define the headers since the data does not have any
headers = ["symboling", "normalized_losses", "make", "fuel_type", "aspiration",
           "num_doors", "body_style", "drive_wheels", "engine_location",
           "wheel_base", "length", "width", "height", "curb_weight",
           "engine_type", "num_cylinders", "engine_size", "fuel_system",
           "bore", "stroke", "compression_ratio", "horsepower", "peak_rpm",
           "city_mpg", "highway_mpg", "price"]

# Read in the CSV file and convert "?" to NaN
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-
                 header=None, names=headers, na_values="?" )
df.head()
```

	ses	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_locatic
	laN	alfa-romero	gas	std	two	convertible	rwd	fro
	laN	alfa-romero	gas	std	two	convertible	rwd	fro
	laN	alfa-romero	gas	std	two	hatchback	rwd	fro
	4.0	audi	gas	std	four	sedan	fwd	fro
	4.0	audi	gas	std	four	sedan	4wd	fro

```
df.dtypes
```

```
symboling             int64
normalized_losses   float64
make                 object
fuel_type            object
aspiration           object
num_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_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
dtype: object
```

```
obj_df = df.select_dtypes(include=['object']).copy()
obj_df.head()
```

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_loca
0	alfa-romero	gas	std	two	convertible	rwd	
1	alfa-romero	gas	std	two	convertible	rwd	
2	alfa-romero	gas	std	two	hatchback	rwd	
3	audi	gas	std	four	sedan	fwd	

```
obj_df[obj_df.isnull().any(axis=1)]
```

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_loc
27	dodge	gas	turbo	NaN	sedan	fwd	
63	mazda	diesel	std	NaN	sedan	fwd	

```
obj_df["num_doors"].value_counts()
```

```
four    114
two      89
Name: num_doors, dtype: int64
```

```
obj_df = obj_df.fillna({"num_doors": "four"})
```

1 - Find and Replace

```
obj_df["num_cylinders"].value_counts()
```

```
four      159
six        24
five       11
eight       5
two         4
three       1
```

```
twelve      1
Name: num_cylinders, dtype: int64
```

```
cleanup_nums = {"num_doors":     {"four": 4, "two": 2},
                "num_cylinders": {"four": 4, "six": 6, "five": 5, "eight": 8,
                                  "two": 2, "twelve": 12, "three":3 }}
```

```
obj_df = obj_df.replace(cleanup_nums)
obj_df.head()
```

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_loca
0	alfa-romero	gas	std	2	convertible	rwd	
1	alfa-romero	gas	std	2	convertible	rwd	
2	alfa-romero	gas	std	2	hatchback	rwd	
3	audi	gas	std	4	sedan	fwd	

```
obj_df.dtypes
```

```
make               object
fuel_type          object
aspiration         object
num_doors           int64
body_style         object
drive_wheels       object
engine_location    object
engine_type        object
num_cylinders       int64
fuel_system        object
dtype: object
```

2 - Label Encoding or Ordinal Encoding

```
obj_df["body_style"] = obj_df["body_style"].astype('category')
obj_df.dtypes
```

```
make                 object
fuel_type            object
aspiration           object
num_doors             int64
body_style         category
drive_wheels         object
engine_location      object
engine_type          object
num_cylinders         int64
fuel_system          object
dtype: object
```

```
obj_df["body_style_cat"] = obj_df["body_style"].cat.codes
obj_df.head()
```

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_loca
0	alfa-romero	gas	std	2	convertible	rwd	
1	alfa-romero	gas	std	2	convertible	rwd	
2	alfa-romero	gas	std	2	hatchback	rwd	
3	audi	gas	std	4	sedan	fwd	

3 - One Hot Encoding

```
pd.get_dummies(obj_df, columns=["drive_wheels"]).head()
```

e	aspiration	num_doors	body_style	engine_location	engine_type	num_cylinders	f
s	std	2	convertible	front	dohc	4	
s	std	2	convertible	front	dohc	4	
s	std	2	hatchback	front	ohcv	6	
s	std	4	sedan	front	ohc	4	
s	std	4	sedan	front	ohc	5	

4 - Custom Binary Encoding

```
obj_df["engine_type"].value_counts()
```

```
ohc      148
ohcf      15
ohcv      13
dohc      12
l         12
rotor      4
dohcv      1
Name: engine_type, dtype: int64
```

```
obj_df["OHC_Code"] = np.where(obj_df["engine_type"].str.contains("ohc"), 1, 0)
```

```
obj_df[["make", "engine_type", "OHC_Code"]].head()
```

	make	engine_type	OHC_Code
0	alfa-romero	dohc	1
1	alfa-romero	dohc	1
2	alfa-romero	ohcv	1
3	audi	ohc	1
4	audi	ohc	1

Scikit-Learn

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
obj_df[["body_style"]] = le.fit_transform(obj_df[["body_style"]])
obj_df[["body_style"]].head()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_label.py:115: DataCo
  y = column_or_1d(y, warn=True)
```

	body_style
0	0
1	0
2	2
3	3
4	3

```
from sklearn.preprocessing import OrdinalEncoder

ord_enc = OrdinalEncoder()
obj_df["make_code"] = ord_enc.fit_transform(obj_df[["make"]])
obj_df[["make", "make_code"]].head()
```

	make	make_code
0	alfa-romero	0.0
1	alfa-romero	0.0
2	alfa-romero	0.0
3	audi	1.0
4	audi	1.0

```
from sklearn.preprocessing import OneHotEncoder
```

```
oe_style = OneHotEncoder()
oe_results = oe_style.fit_transform(obj_df[["body_style"]])
pd.DataFrame(oe_results.toarray(), columns=oe_style.categories_).head()
```

	convertible	hardtop	hatchback	sedan	wagon
0	1.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0
3	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	1.0	0.0

Advanced Approaches

```
!pip install category_encoders


#Backward Difference encoding.
import category_encoders as ce

# Get a new clean dataframe
obj_df = df.select_dtypes(include=['object']).copy()

# Specify the columns to encode then fit and transform
encoder = ce.BackwardDifferenceEncoder(cols=["engine_type"])
encoder.fit_transform(obj_df, verbose=1).iloc[:,8:14].head()
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWar
  import pandas.util.testing as tm
```

	engine_type_0	engine_type_1	engine_type_2	engine_type_3	engine_type_4	eng
0	-0.857143	-0.714286	-0.571429	-0.428571	-0.285714	
1	-0.857143	-0.714286	-0.571429	-0.428571	-0.285714	
2	0.142857	-0.714286	-0.571429	-0.428571	-0.285714	
3	0.142857	0.285714	-0.571429	-0.428571	-0.285714	
4	0.142857	0.285714	-0.571429	-0.428571	-0.285714	

```
#If we try a polynomial encoding, we get a different distribution of values used to encode

encoder = ce.PolynomialEncoder(cols=["engine_type"])
encoder.fit_transform(obj_df, verbose=1).iloc[:,8:14].head()
```

	engine_type_0	engine_type_1	engine_type_2	engine_type_3	engine_type_4	eng
0	-0.566947	5.455447e-01	-0.408248	0.241747	-0.109109	
1	-0.566947	5.455447e-01	-0.408248	0.241747	-0.109109	
2	-0.377964	9.521795e-17	0.408248	-0.564076	0.436436	
3	-0.188982	-3.273268e-01	0.408248	0.080582	-0.545545	

```
encoder = ce.BinaryEncoder(cols=["engine_type"])
encoder.fit_transform(obj_df).head()
```

	make	fuel_type	aspiration	num_doors	body_style	drive_wheels	engine_loca
0	alfa-romero	gas	std	two	convertible	rwd	
1	alfa-romero	gas	std	two	convertible	rwd	
2	alfa-romero	gas	std	two	hatchback	rwd	
3	audi	gas	std	four	sedan	fwd	
4	audi	gas	std	four	sedan	4wd	

✓ 0s completed at 12:58 PM