```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

import tensorflow as tf
from tensorflow import keras
from keras import layers

import warnings
warnings.filterwarnings('ignore')
```

```
df = pd.read_csv('/content/auto-mpg.csv')
df.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	ca nar
0	18.0	8	307.0	130	3504	12.0	70	1	chevrol cheve malil
1	15.0	8	350.0	165	3693	11.5	70	1	bui skyla 32

```
df.shape
```

```
(398, 9)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
```

```
 8   car name      398 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

```
df.describe()
```

	mpg	cylinders	displacement	weight	acceleration	model year	or
count	398.000000	398.000000	398.000000	398.000000	398.000000	398.000000	398.00
mean	23.514573	5.454774	193.425879	2970.424623	15.568090	76.010050	1.57
std	7.815984	1.701004	104.269838	846.841774	2.757689	3.697627	0.80
min	9.000000	3.000000	68.000000	1613.000000	8.000000	70.000000	1.00
25%	17.500000	4.000000	104.250000	2223.750000	13.825000	73.000000	1.00
50%	23.000000	4.000000	148.500000	2803.500000	15.500000	76.000000	1.00
75%	29.000000	8.000000	262.000000	3608.000000	17.175000	79.000000	2.00

```
df['horsepower'].unique()
```

```
array(['130', '165', '150', '140', '198', '220', '215', '225', '190',
       '170', '160', '95', '97', '85', '88', '46', '87', '90', '113',
       '200', '210', '193', '?', '100', '105', '175', '153', '180', '110',
       '72', '86', '70', '76', '65', '69', '60', '80', '54', '208', '155',
       '112', '92', '145', '137', '158', '167', '94', '107', '230', '49',
       '75', '91', '122', '67', '83', '78', '52', '61', '93', '148',
       '129', '96', '71', '98', '115', '53', '81', '79', '120', '152',
       '102', '108', '68', '58', '149', '89', '63', '48', '66', '139',
       '103', '125', '133', '138', '135', '142', '77', '62', '132', '84',
       '64', '74', '116', '82'], dtype=object)
```

```
print(df.shape)
df = df[df['horsepower'] != '?']
print(df.shape)
```

```
(398, 9)
(392, 9)
```

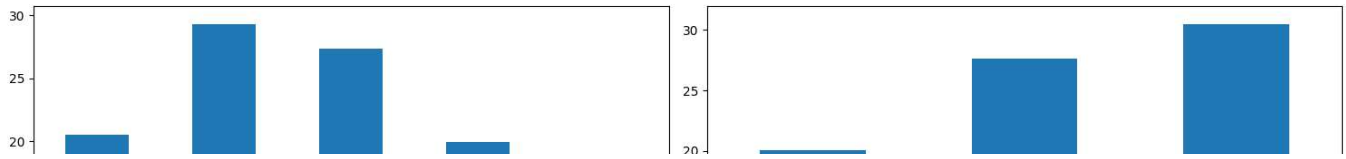
```
df['horsepower'] = df['horsepower'].astype(int)
df.isnull().sum()
```

```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

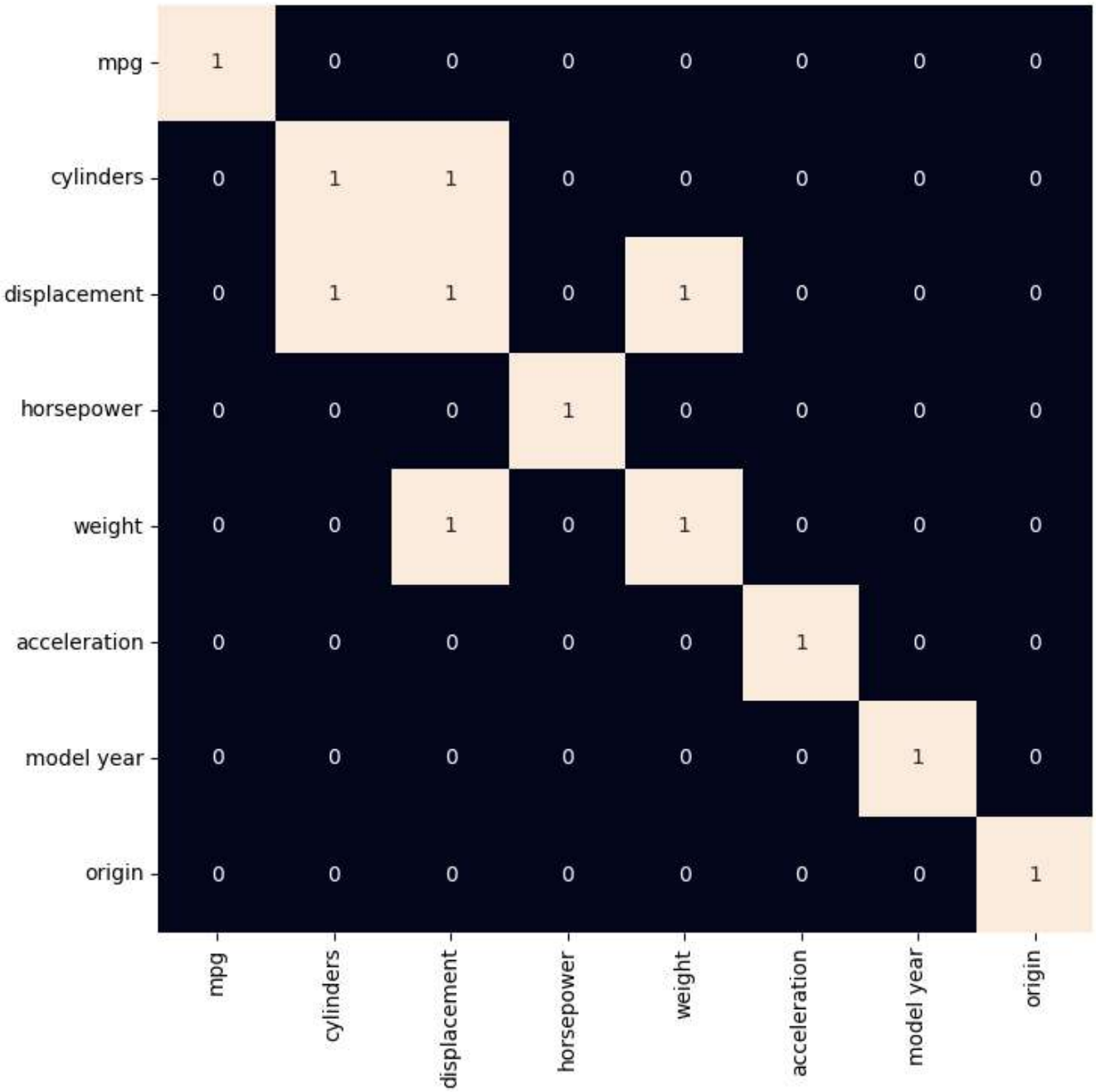
```
df.nunique()
```

```
mpg             127
cylinders         5
displacement     81
horsepower       93
weight          346
acceleration     95
model year       13
origin            3
car name        301
dtype: int64
```

```
plt.subplots(figsize=(15, 5))
for i, col in enumerate(['cylinders', 'origin']):
    plt.subplot(1, 2, i+1)
    x = df.groupby(col).mean()['mpg']
    x.plot.bar()
    plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(8, 8))
sb.heatmap(df.corr() > 0.9,
           annot=True,
           cbar=False)
plt.show()
```



```
df.drop('displacement',
        axis=1,
        inplace=True)
```

```
from sklearn.model_selection import train_test_split
features = df.drop(['mpg', 'car name'], axis=1)
target = df['mpg'].values

X_train, X_val, \
    Y_train, Y_val = train_test_split(features, target,
                                      test_size=0.2,
                                      random_state=22)
X_train.shape, X_val.shape
```

```
((313, 6), (79, 6))
```

```
AUTO = tf.data.experimental.AUTOTUNE

train_ds = (
    tf.data.Dataset
    .from_tensor_slices((X_train, Y_train))
    .batch(32)
    .prefetch(AUTO)
)

val_ds = (
    tf.data.Dataset
    .from_tensor_slices((X_val, Y_val))
    .batch(32)
    .prefetch(AUTO)
)
```

```
model = keras.Sequential([
    layers.Dense(256, activation='relu', input_shape=[6]),
    layers.BatchNormalization(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(1, activation='relu')
])
```

```
model.compile(
    loss='mae',
    optimizer='adam',
```

```
    metrics=['mape']
)
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               1792

 batch_normalization (Batch  (None, 256)               1024
 Normalization)

 dense_1 (Dense)             (None, 256)               65792

 dropout (Dropout)           (None, 256)               0

 batch_normalization_1 (Bat  (None, 256)               1024
 chNormalization)

 dense_2 (Dense)             (None, 1)                 257

=================================================================
Total params: 69889 (273.00 KB)
Trainable params: 68865 (269.00 KB)
Non-trainable params: 1024 (4.00 KB)
_________________________________________________________________
```

```
history = model.fit(train_ds,
                    epochs=50,
                    validation_data=val_ds)
```

```
Epoch 1/50
10/10 [==============================] - 1s 16ms/step - loss: 21.3930 - mape: 85.888
Epoch 2/50
10/10 [==============================] - 0s 5ms/step - loss: 19.6603 - mape: 75.3603
Epoch 3/50
10/10 [==============================] - 0s 3ms/step - loss: 19.1885 - mape: 73.2051
Epoch 4/50
10/10 [==============================] - 0s 4ms/step - loss: 18.5571 - mape: 69.9519
Epoch 5/50
10/10 [==============================] - 0s 3ms/step - loss: 17.9341 - mape: 66.8491
Epoch 6/50
10/10 [==============================] - 0s 3ms/step - loss: 17.4150 - mape: 64.5791
Epoch 7/50
10/10 [==============================] - 0s 3ms/step - loss: 16.9497 - mape: 62.4375
Epoch 8/50
10/10 [==============================] - 0s 3ms/step - loss: 16.3605 - mape: 59.8049
Epoch 9/50
```

```
10/10 [==============================] - 0s 4ms/step - loss: 15.6625 - mape: 56.6876
Epoch 10/50
10/10 [==============================] - 0s 4ms/step - loss: 15.4228 - mape: 56.2952
Epoch 11/50
10/10 [==============================] - 0s 3ms/step - loss: 14.7401 - mape: 52.9114
Epoch 12/50
10/10 [==============================] - 0s 5ms/step - loss: 14.0628 - mape: 50.1103
Epoch 13/50
10/10 [==============================] - 0s 6ms/step - loss: 13.5715 - mape: 48.5248
Epoch 14/50
10/10 [==============================] - 0s 7ms/step - loss: 12.8673 - mape: 45.1738
Epoch 15/50
10/10 [==============================] - 0s 6ms/step - loss: 12.4747 - mape: 44.3483
Epoch 16/50
10/10 [==============================] - 0s 5ms/step - loss: 11.8209 - mape: 41.5041
Epoch 17/50
10/10 [==============================] - 0s 5ms/step - loss: 11.4732 - mape: 40.7197
Epoch 18/50
10/10 [==============================] - 0s 5ms/step - loss: 11.3021 - mape: 40.1263
Epoch 19/50
10/10 [==============================] - 0s 5ms/step - loss: 11.0246 - mape: 39.3251
Epoch 20/50
10/10 [==============================] - 0s 5ms/step - loss: 10.7325 - mape: 38.0912
Epoch 21/50
10/10 [==============================] - 0s 5ms/step - loss: 10.5785 - mape: 38.0451
Epoch 22/50
10/10 [==============================] - 0s 5ms/step - loss: 10.1575 - mape: 36.1645
Epoch 23/50
10/10 [==============================] - 0s 5ms/step - loss: 10.0571 - mape: 36.1526
Epoch 24/50
10/10 [==============================] - 0s 5ms/step - loss: 9.8574 - mape: 35.4619
Epoch 25/50
10/10 [==============================] - 0s 5ms/step - loss: 9.4007 - mape: 34.0875
Epoch 26/50
10/10 [==============================] - 0s 5ms/step - loss: 9.0616 - mape: 33.0060
Epoch 27/50
10/10 [==============================] - 0s 5ms/step - loss: 8.6498 - mape: 30.9073
Epoch 28/50
10/10 [==============================] - 0s 5ms/step - loss: 8.1784 - mape: 29.9738
```

```
history_df = pd.DataFrame(history.history)
history_df.head()
```

```
              loss      mape  val_loss  val_mape
history_df.loc[:, ['loss', 'val_loss']].plot()
history_df.loc[:, ['mape', 'val_mape']].plot()
plt.show()
```