Import DataSet avilable at given url https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data

21162101007_KshitijGupta

Suggested code may be subject to a license | jrtorresb/ML

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd

# Define the path to the dataset
path = "https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data"

# Load the dataset into a pandas dataframe
df = pd.read_csv(path, na_values="?", header=None)

# Define the headers for the dataframe
headers = [
    "symboling",
    "normalized-losses",
    "make",
    "fuel-type",
    "aspiration",
    "num-of-doors",
    "body-style",
    "drive-wheels",
    "engine-location",
    "wheel-base",
    "length",
    "width",
    "height",
    "curb-weight",
    "engine-type",
    "num-of-cylinders",
    "engine-size",
    "fuel-system",
    "bore",
    "stroke",
    "compression-ratio",
    "horsepower",
    "peak-rpm",
    "city-mpg",
    "highway-mpg",
    "price"
]

# Assign the headers to the dataframe columns
df.columns = headers

# Display the first 5 rows of the dataframe
print("The first 5 rows of the dataframe:")
print(df.head())
```

```
The first 5 rows of the dataframe:
   symboling  normalized-losses         make fuel-type aspiration  \
0          3                NaN  alfa-romero       gas        std
1          3                NaN  alfa-romero       gas        std
2          1                NaN  alfa-romero       gas        std
3          2              164.0         audi       gas        std
4          2              164.0         audi       gas        std

  num-of-doors   body-style drive-wheels engine-location  wheel-base  ...  \
0          two  convertible          rwd           front        88.6  ...
1          two  convertible          rwd           front        88.6  ...
2          two    hatchback          rwd           front        94.5  ...
3         four        sedan          fwd           front        99.8  ...
4         four        sedan          4wd           front        99.4  ...

   engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0          130         mpfi  3.47    2.68               9.0      111.0
1          130         mpfi  3.47    2.68               9.0      111.0
2          152         mpfi  2.68    3.47               9.0      154.0
3          109         mpfi  3.19    3.40              10.0      102.0
4          136         mpfi  3.19    3.40               8.0      115.0
```

```
   peak-rpm city-mpg  highway-mpg    price
0    5000.0       21           27  13495.0
1    5000.0       21           27  16500.0
2    5000.0       19           26  16500.0
3    5500.0       24           30  13950.0
4    5500.0       18           22  17450.0

[5 rows x 26 columns]
```

Data PreProsseing

```
import numpy as np
import pandas as pd

# Replace missing values with NaN
df['normalized-losses'].replace('?', np.nan, inplace=True)
df['bore'].replace('?', np.nan, inplace=True)
df['stroke'].replace('?', np.nan, inplace=True)
df['horsepower'].replace('?', np.nan, inplace=True)
df['peak-rpm'].replace('?', np.nan, inplace=True)
df['num-of-doors'].replace('?', np.nan, inplace=True)

# Fill missing values with the mean for numerical columns
df['normalized-losses'].fillna(df['normalized-losses'].astype('float').mean(), inplace=True)
df['bore'].fillna(df['bore'].astype('float').mean(), inplace=True)
df['stroke'].fillna(df['stroke'].astype('float').mean(), inplace=True)
df['horsepower'].fillna(df['horsepower'].astype('float').mean(), inplace=True)
df['peak-rpm'].fillna(df['peak-rpm'].astype('float').mean(), inplace=True)

# Fill missing values with the mode for categorical columns
df['num-of-doors'].fillna(df['num-of-doors'].mode()[0], inplace=True)

# Convert data types to appropriate formats
df['price'] = df['price'].replace('?', np.nan).astype('float')
df.dropna(subset=['price'], inplace=True)  # Remove rows with NaN values in 'price'
df['price'] = df['price'].astype('float')
df['normalized-losses'] = df['normalized-losses'].astype('float')
df['bore'] = df['bore'].astype('float')
df['stroke'] = df['stroke'].astype('float')
df['horsepower'] = df['horsepower'].astype('float')
df['peak-rpm'] = df['peak-rpm'].astype('float')
```

List Down All the Continuous Attributes in the dataset

```
# Identify continuous attributes in the dataframe
continuous_attributes = df.select_dtypes(include=['float64', 'int64']).columns.tolist()

# Print continuous attributes
print("\nContinuous Attributes: \n")
for attribute in continuous_attributes:
    print(f' --> {attribute}')
```

```
Continuous Attributes:

 --> symboling
 --> normalized-losses
 --> wheel-base
 --> length
 --> width
 --> height
 --> curb-weight
 --> engine-size
 --> bore
 --> stroke
 --> compression-ratio
 --> horsepower
 --> peak-rpm
 --> city-mpg
 --> highway-mpg
 --> price
```

List Down all the Categorical attributes in the dataset

```
# Identify categorical attributes in the dataframe
categorical_attributes = df.select_dtypes(include=['object']).columns.tolist()

# Print categorical attributes
print("\nCategorical Attributes: \n")
for attribute in categorical_attributes:
    print(f' --> {attribute}')
```

```
Categorical Attributes:

 --> make
 --> fuel-type
 --> aspiration
 --> num-of-doors
 --> body-style
 --> drive-wheels
 --> engine-location
 --> engine-type
 --> num-of-cylinders
 --> fuel-system
```

Draw reglot between each continuous attribute and price and write down whether that attribute is related to price or not

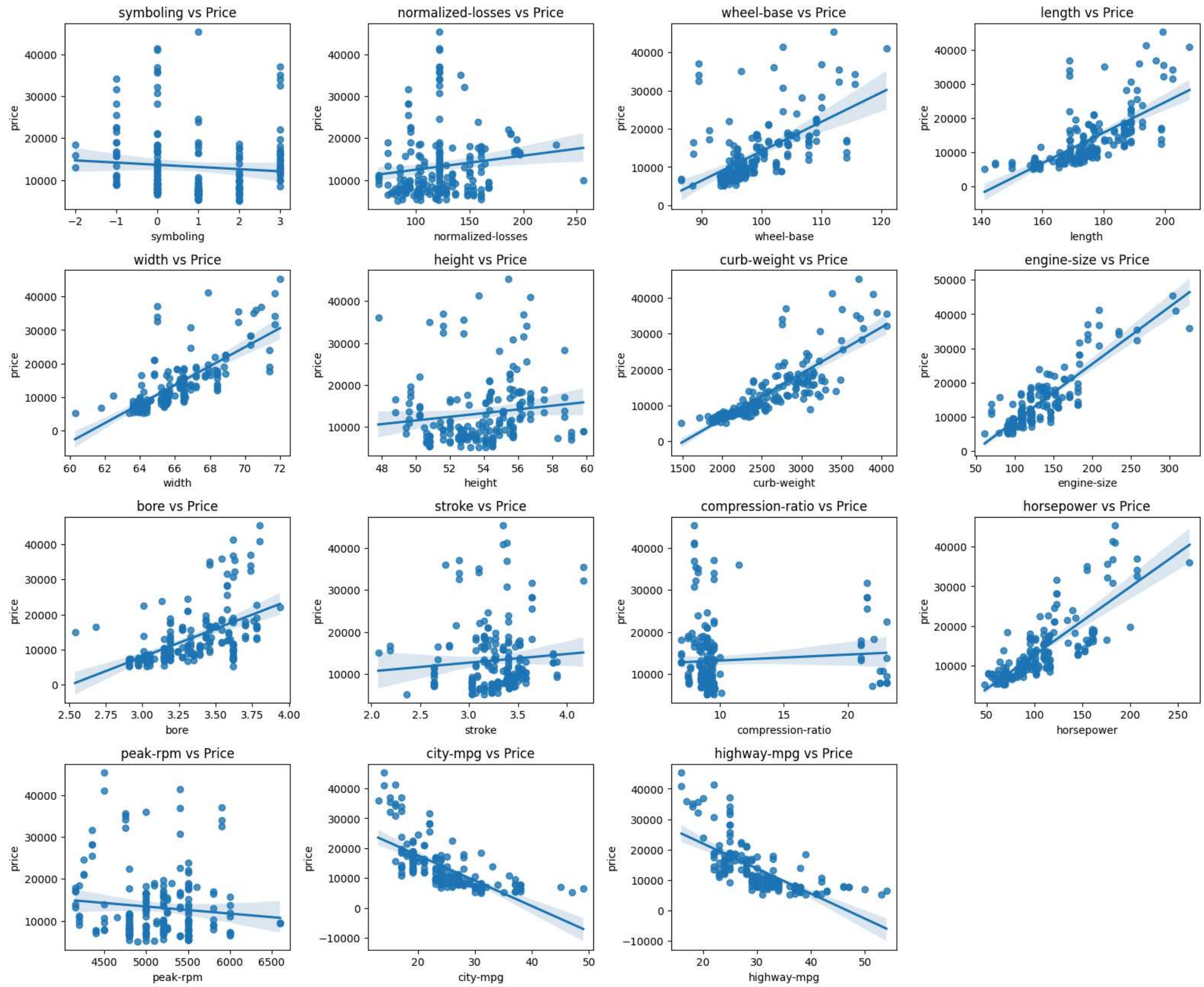
Generate print hello world using rot13 Close

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

# Data visualization for continuous attributes
plt.figure(figsize=(16, 16))
for i, attr in enumerate(continuous_attributes):
    if attr != 'price':
        plt.subplot(5, 4, i+1)
        sns.regplot(x=attr, y='price', data=df)
        plt.title(f'{attr} vs Price')
        plt.tight_layout()

plt.show()

# Check if continuous attributes are related to price
print("\nRelationship of Continuous Attributes with Price:\n")
for attr in continuous_attributes:
    if attr != 'price':
        correlation = df[[attr, 'price']].corr().iloc[0, 1]
        print(f'{attr}: {"✓" if abs(correlation) > 0.5 else "X"}')
```




```
Relationship of Continuous Attributes with Price:

symboling: X
normalized-losses: X
wheel-base: ✓
length: ✓
width: ✓
height: X
curb-weight: ✓
engine-size: ✓
bore: ✓
stroke: X
compression-ratio: X
horsepower: ✓
peak-rpm: X
city-mpg: ✓
highway-mpg: ✓
```

Draw Boxplot between each categorical attribute and price and write down whether that attribute is related to price or not.

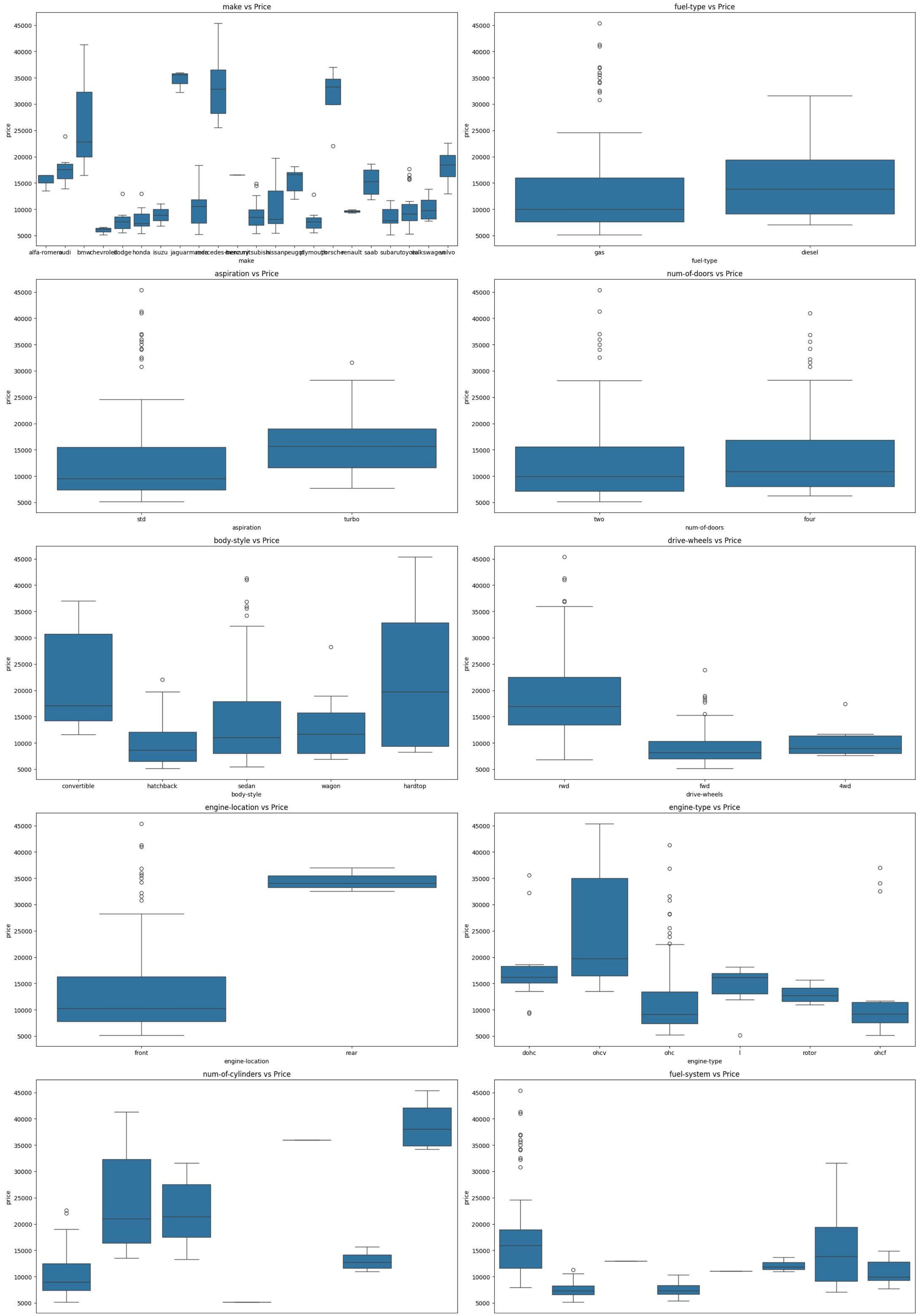
```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Data visualization for categorical attributes
plt.figure(figsize=(22, 32))
for i, attr in enumerate(categorical_attributes):
    plt.subplot(5, 2, i+1)
    sns.boxplot(x=attr, y='price', data=df)
    plt.title(f'{attr} vs Price')
    plt.tight_layout()

plt.show()

# Check if categorical attributes are related to price
print("\nRelationship of Categorical Attributes with Price:\n")
for attr in categorical_attributes:
    grouped_test = df[[attr, 'price']].groupby([attr])
    unique_values = df[attr].unique()

    if len(unique_values) > 1:
        f_val, p_val = stats.f_oneway(*[grouped_test.get_group(val)['price'] for val in unique_values if val in grouped_test.groups])
        print(f'{attr}: {"✓" if p_val < 0.05 else "X"}')
```


make vs Price
price
make
fuel-type vs Price
price
gas
diesel
fuel-type
aspiration vs Price
price
std
turbo
aspiration
num-of-doors vs Price
price
two
four
num-of-doors
body-style vs Price
price
convertible
hatchback
sedan
wagon
hardtop
body-style
drive-wheels vs Price
price
rwd
fwd
4wd
drive-wheels
engine-location vs Price
price
front
rear
engine-location
engine-type vs Price
price
dohc
ohcv
ohc
l
rotor
ohcf
engine-type
num-of-cylinders vs Price
price
fuel-system vs Price
price


```
Relationship of Categorical Attributes with Price:

make: ✓
fuel-type: X
aspiration: ✓
num-of-doors: X
body-style: ✓
drive-wheels: ✓
engine-location: ✓
engine-type: ✓
num-of-cylinders: ✓
fuel-system: ✓
```

Calculate pearson correlation between each continuous attribute and price and write down whether that attribute is related to price or not.

```
print("\nPearson Correlation Coefficients with Price (and if related):\n")
for attr in continuous_attributes:
    if attr != 'price':
        correlation = df[[attr, 'price']].corr().iloc[0, 1]
        print(f'{attr}: {correlation:.2f} {"(✓)" if abs(correlation) > 0.5 else "(X)"}')
```

```
Pearson Correlation Coefficients with Price (and if related):

symboling: -0.08 (X)
normalized-losses: 0.13 (X)
wheel-base: 0.58 (✓)
length: 0.69 (✓)
width: 0.75 (✓)
height: 0.14 (X)
curb-weight: 0.83 (✓)
engine-size: 0.87 (✓)
```