```
# Exploratory Data Analysis on Automobile Dataset

# Import relevant libraries and define settings for plotting.

import numpy as np # for numerical function

import pandas as pd # for data preprocessing

import matplotlib.pyplot as plt # for data visualization

%matplotlib inline

import seaborn as sns

import datetime

import time

# To Ignore Warning

import warnings

warnings.filterwarnings("ignore")

# Data Loading

df = pd.read_csv('Automobile_data.csv')

df.head()

#Summary statistics of variable

df.describe()

#To check datatypes of all column

df.info()

df.dtypes

df.shape

df['losses'].unique() # ? is missing value

# Data Cleaning

First Step

Data contains "?" replace it with NAN

df = df.replace('?',np.NAN)

df = df.replace('#',np.NAN)

df = df.replace('$',np.NAN)

df.isnull().sum()
```

fill missing data of normalised-losses, price, horsepower, peak-rpm, bore, stroke with the respective column mean

Fill missing data category Number of doors with the mode of the column i.e. Four

Second Step

```
print(df['losses'].astype('str').astype('float').mean(axis=0))


avg_losses = df['losses'].astype('str').astype('float').mean(axis=0)

df['losses'].replace(np.nan, avg_losses, inplace=True)

print(df['losses'].astype('int'))

df.columns

df['losses'].unique()

df['total price'].unique() # ? missing value


avg_price = df['total price'].astype('float').mean(axis=0)

df['total price'].replace(np.nan, avg_price, inplace=True)

print(df['total price'].astype('int'))

df['total price'].unique()

df.columns

df['horsepower'].unique()

avg_horsepower = df['horsepower'].astype('str').astype('float').mean(axis=0)

df['horsepower'].replace(np.nan, avg_horsepower, inplace=True)

print(df['horsepower'].astype('int'))

df['horsepower'].unique()

avg_peak_rpm = df['peak-rpm'].astype('str').astype('float').mean(axis=0)

df['peak-rpm'].replace(np.nan, avg_peak_rpm, inplace=True)

print(df['peak-rpm'].astype('int'))

df.head(20)

# Replace the non-numeric value to null and convert the datatype
```

```
df['bore'] = pd.to_numeric(df['bore'],errors='coerce')

df.dtypes

# Replace the non-number value to null and convert the datatype

df['stroke'] = pd.to_numeric(df['stroke'],errors='coerce')

df.dtypes

# Convert the non-numeric data to null and convert the datatype

df['peak-rpm'] = pd.to_numeric(df['peak-rpm'],errors='coerce')

df.dtypes

df.isnull().sum()

df = df.dropna()

df.isnull().sum()

df.shape

# Univariate Analysis
```

Univariate analysis is the simplest form of analyzing data. “Uni” means “one”, so in other words your data has only one variable. It doesn't deal with causes or relationships (unlike regression) and it's major purpose is to describe; It takes data, summarizes that data and finds patterns in the data.

```
# Vehicle by  make frequency diagram
```

by this we say that toyota make more vehical

```
df.make.value_counts().nlargest(10).plot(kind='bar', figsize=(15,5))

plt.title("Number of vehicles by make")

plt.ylabel('Number of vehicles')

plt.xlabel('Company')

# symboling Histogram
```

from this histogram we say that mostlt symbloing values are between 0 to 2

df.hist(column='risk factor',bins=6,color='orange')

Normalized losses histogram

normalized-losses mostly range between 75 to 125

df.hist(column='losses',bins=6,color='orange')

histogram for all columns

df.hist(figsize=(20,30))

Bar plot for Fuel type

from this we can find count of fuel-typ

sns.countplot(x="type of fuel", data=df)

fuel = ['gas', 'diesel']

data = df["type of fuel"].value_counts()

explode = (0.1, 0.0)

fig = plt.figure(figsize =(15, 8))

colors = ['#B7C3F3','#4F4372']

plt.pie(data, labels = fuel,explode=explode,autopct='%1.2f%%', shadow=True,colors=colors)

plt.title("Fuel Types")

aspiration_types = ['std', 'turbo']

data = df["aspiration"].value_counts()

explode = (0.1, 0.0)

fig = plt.figure(figsize =(15, 8))

colors = ['#B7C3F3','#4F4372']

plt.pie(data, labels = aspiration_types,explode=explode,autopct='%1.2f%%', shadow=True,colors=colors)

plt.title("Aspiration Dominance")

Bar plot for drive wheels

sns.countplot(x="wheels", data=df)

Bar plot for num of doors

sns.countplot(x="total doors", data=df)

16. Name 2 most frequently occurring cars by “body”.

sns.countplot(x="body", data=df)

find the correlation of columns with each other

corr = df.corr()

corr

plot heatmap for corr

find with column are more correlated with price

#19. What are the most important features in the dataset that are correlated with the target variable, and how strong are these correlations?

sns.heatmap(corr, annot=True,)

sns.countplot(x="type of fuel", data=df)

Bivariate Analysis

Bivariate analysis is one of the simplest forms of quantitative (statistical) analysis. It involves the analysis of two variables (often denoted as X, Y), for the purpose of determining the empirical relationship between them.

#set a fig size

plt.rcParams['figure.figsize']=(30,15)

make a boxplot for make and price

Findings: Below are our findings on the make and price of the car

- The most expensive car is manufacture by Mercedes benz and the least expensive is Chevrolet
- The premium cars costing more than 20000 are BMW, Jaquar, Mercedes benz and Porsche
- Less expensive cars costing less than 10000 are Chevrolet, Dodge, Honda, Mitsubishi, Plymoth and Subaru
- Rest of the cars are in the midrange between 10000 and 20000 which has the highest number of cars

```
print(df['total price'].astype('str'))
print(df['make'].astype('str'))
sns.boxplot(x=df["total price"], y=df["make"])
# make a boxplot for drive-wheel and price
sns.boxplot(x='wheels', y='total price', data=df)
# display a distrubation of price
```

so this this distrubation is A "skewed right" distribution.in which the tail is on the right side.

A positive skewness indicates that the size of the right-handed tail is larger than the left-handed tail.

skewness affect on data:

mean greater than the mode,

median greater than the mode,

mean greater than median,

Here we se that most of car price are in between 5000 to 20000 and there is rare cars which are expensive

The kurtosis parameter is a measure of the combined weight of the tails relative to the rest of the distribution.”

if kurtosis is positive than it shows a dataset with more weight in the tails.

```
sns.distplot(df['total price'])
print('This distribution has skew', df['total price'].skew())
print('This distribution has kurtosis', df['total price'].kurt())
```

Distrubation of length

sns.distplot(df['length'])

distrubation of highway-mpg

sns.distplot(df['mpg on highway'])

Scatter plot of price and engine size

The more the engine size the costlier the price is

sns.lmplot(x='total price', y='size of engine', data=df)

plt.show()

Scatter plot of price and highway-mpg

The low the highw-mpg cheap the price is

df.plot.scatter(x = 'total cylinders', y = 'size of engine', c = 'red');

print(df['losses'].astype('float'))

print(df['total doors'].astype('str'))

#14. Does cars with different number of doors have different losses? Explain with the help of suitable plot.

df.groupby('total doors')['losses'].mean().plot(kind='barh', color = 'blue')

plt.title("Risk factor by total doors")

plt.ylabel('total doors')

plt.xlabel('losses')

15. Does cars having 4-doors have more safety than others, explain using column “risk factor”?

df.groupby('total doors')['risk factor'].mean().plot(kind='barh', color = 'blue')

plt.title("Risk factor by total doors")

plt.ylabel('total doors')

plt.xlabel('risk factor')

print(df['losses'].astype('int'))

print(df['aspiration'].astype('str'))

12. Out of which aspiration () losses are higher? Use histogram to answer this.

#ax = df.plot.hist(column=['losses'], by="aspiration", figsize=(10, 8))

```
df.groupby('aspiration')['losses'].mean().plot(kind='barh', color = 'blue')

plt.title("Risk factor by total doors")

plt.ylabel('aspiration')

plt.xlabel('losses')

sns.lmplot('price',"highway-mpg", df)

# Scatter plot of City and Highway MPG, Curb weight based on Make of the car
```

Heavier the Automobile less is the mileage for both City and Highway

```
sns.lmplot('highway-mpg',"curb-weight", df, hue="make",fit_reg=False)

sns.lmplot('city-mpg',"curb-weight", df, hue="make", fit_reg=False)

# Drive wheels and City MPG bar chart


df.groupby('wheels')['mpg in city'].mean().plot(kind='barh', color = 'blue')

plt.title("Drive wheels City MPG")

plt.ylabel('City MPG')

plt.xlabel('Drive wheels')

# Drive wheels and Highway MPG bar chart

df.groupby('wheels')['mpg on highway'].mean().plot(kind='bar', color = 'pink');

plt.title("Drive wheels Highway MPG")

plt.ylabel('Highway MPG')

plt.xlabel('Drive wheels')


# Normalized losses based on body style and no. of doors
```

Findings: As we understand the normalized loss which is the average loss payment per insured vehicle is calculated with many features of the cars which includes body style and no. of doors. Normalized losses are distributed across different body style but the two door cars has more number of losses than the four door cars.

```
pd.pivot_table(df,index=['body','total doors'], values='losses').plot(kind='bar',color='orange')

plt.title("Normalized losses based on body style and no. of doors")

plt.ylabel('Normalized losses')
```

```
plt.xlabel('Body style and No. of doors')
dummy_variable_1 = pd.get_dummies(df["type of fuel"])
dummy_variable_1.head()

dummy_variable_1.rename(columns={'gas':'fuel-type-gas', 'diesel':'fuel-type-diesel'}, inplace=True)

dummy_variable_1.head()
df = pd.concat([df, dummy_variable_1], axis=1)
# drop original column "fuel-type" from "df"

df.drop("type of fuel", axis = 1, inplace=True)

df.head()
dummy_variable_2 = pd.get_dummies(df['aspiration'])

dummy_variable_2.rename(columns={'std':'aspiration-std', 'turbo': 'aspiration-turbo'}, inplace=True)

dummy_variable_2.head()
# Merging the new dataframe to the original dataframe, then drop the column 'aspiration'.

df = pd.concat([df, dummy_variable_2], axis=1)

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

df.head()
#Save the new csv:

df.to_csv('clean_df.csv')
# we can calculate the correlation between variables
```

# of type "int64" or "float64" using the method "corr":

df.corr()

df[['bore', 'stroke', 'compression-ratio', 'horsepower']].corr()

#14. Does cars with different number of doors have different losses? Explain with the help of suitable plot.

sns.regplot(x="total doors", y="losses", data=df)

plt.ylim(0,)

#12. Out of which aspiration () losses are higher? Use histogram to answer this.

sns.regplot(x="aspiration", y="losses", data=df)

plt.ylim(0,)

df[["size of engine", "total price"]].corr()

#23. What is the performance of different baseline models on the dataset, such as linear regression or decision trees?

df.rename(columns={'total price': 'price'}, inplace=True)

df['price'].fillna((df['price'].astype('float').mean()), inplace=True)

#df.price.fillna(value=0, inplace=True)

X = df[['mpg on highway']]

Y = df['price']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)

from sklearn.linear_model import LinearRegression

#2) Create a Linear regression model between Features and target data

model_li = LinearRegression()

model_li.fit(X_train,y_train)

model_li.score(X_train,y_train)

model_li.score(X_test,y_test)

y_pred = model_li.predict(X_test)

from sklearn.preprocessing import PolynomialFeatures

pr=PolynomialFeatures(degree=2)

pr

Z = df[['horsepower', 'weight of curb', 'size of engine', 'mpg on highway']]

#21. Are there any patterns or clusters in the data that can be visualized using unsupervised learning techniques like clustering or dimensionality reduction?

plt.figure(figsize=(8,8))

explode=(0.1,0.05,0.05)

df['make'].value_counts().plot.pie(autopct='%1.1f%%',startangle=60)

plt.title('Make')

del df['losses']

df['total cylinders'].value_counts()

df['total cylinders'] = df['total cylinders'].map({'twelve':1, 'three':1, 'two':1, 'eight':2, 'five':3,'six':4,'four':5})

df['type of fuel'].value_counts()

df['type of fuel'] = df['type of fuel'].map({'gas':2,'diesel':1})

df['aspiration'].value_counts()

df = pd.get_dummies(df, drop_first=True)

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

%matplotlib inline

```
import warnings
warnings.filterwarnings("ignore")
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
def cluster(x,clusters):
    allscore=[]
    allclusters=[]
    sum_of_squared_distances = []
    x=x
    for i in np.arange(1,clusters):

        i+=1
        model=KMeans(n_clusters=i)
        pred=model.fit_predict(x)
        s_score = silhouette_score(x,pred)
        score=silhouette_score(x,pred)
        print("number of cluster {}, silhouette {}".format(i,score))
        allscore.append(s_score)
        allclusters.append(i)
        sum_of_squared_distances.append(model.inertia_)

    plt.plot(allclusters,sum_of_squared_distances, marker='x')
    plt.xlabel('k')
    plt.ylabel('Distortion')
    plt.title('The Elbow Method showing optimal K')
    plt.show()
cluster(df,10)
model = KMeans(n_clusters = 3)
```

```
model = model.fit(df)

pred = model.predict(df)

df['cluster'] = pred

df.head()

df1 = pd.read_csv('Automobile_data.csv')

plt.figure(figsize=(15,15))

plt.scatter(df1['type of fuel'],df1['make'], c=pred)

plt.legend()

plt.xticks(rotation=90)

plt.colorbar()

plt.show()
```