This lab is designed to provide hands-on experience with various machine learning algorithms, tools, and techniques. You'll gain practical skills in data preprocessing, model training, evaluation, and deployment.

The lab is divided into several modules, each focusing on different aspects of machine learning:

1. **Introduction to Machine Learning**
2. **Data Preprocessing and Exploration**
3. **Supervised Learning**
4. **Unsupervised Learning**
5. **Model Evaluation and Validation**
6. **Model Deployment**

CYCLE 1

- Data analysis involves different processes of cleaning, transforming, analyzing the data, and building models to extract specific, relevant insights.
- These are beneficial for making important business decisions in real-time situations. Exploratory Data Analysis is important for any business.
- It lets data scientists analyze the data before reaching any conclusion. Also, this makes sure that the results which are out are valid and applicable to business outcomes and goals.

What is Exploratory Data Analysis in Data Science?

Exploratory Data Analysis (EDA) is one of the techniques used for extracting vital features and trends used by machine learning and deep learning models in Data Science. Thus, EDA has become an important milestone for anyone working in data science

Steps Involved in Exploratory Data Analysis (EDA)

1. Data Collection
2. Finding all Variables and Understanding Them
3. Cleaning the Dataset
4. Identify Correlated Variables
5. Choosing the Right Statistical Methods
6. Visualizing and Analyzing Results

Types of Exploratory Data Analysis

There are three main types of EDA:

1. Univariate
2. Bivariate
3. Multivariate

(1) EDA Univariate Analysis

- Analyzing/visualizing the dataset by taking one variable at a time:
- We visualize our data using Matplotlib and Seaborn libraries.
- Matplotlib is a Python 2D plotting library used to draw basic charts.
- Seaborn is also a python library built on top of Matplotlib that uses short lines of code to create and style statistical plots from Pandas and Numpy.
- Univariate analysis can be done for both Categorical and Numerical variables.
- Categorical variables can be visualized using a Count plot, Bar Chart, Pie Plot, etc.
- Numerical Variables can be visualized using Histogram, Box Plot, Density Plot, etc.

(2) EDA Bivariate Analysis

- Bivariate Analysis helps to understand how variables are related to each other and the relationship between dependent and independent variables present in the dataset.
- For Numerical variables, Pair plots and Scatter plots are widely been used to do Bivariate Analysis.
- A Stacked bar chart can be used for categorical variables if the output variable is a classifier.
- Bar plots can be used if the output variable is continuous

(3) EDA Multivariate Analysis

- Multivariate analysis looks at more than two variables.
- Multivariate analysis is one of the most useful methods to determine relationships and analyze patterns for any dataset.
- **A heat map is widely been used for Multivariate Analysis**
- Heat Map gives the correlation between the variables, whether it has a positive or negative correlation.

1. Data Collection: Read the Data

```
import pandas  as pd

import numpy as np

import matplotlib.pylplot as plt

import seaborn as  sna

#to ignore warnings

Import warnings

Warnings.filterwarinings('ignore')
```

After importing of all the libraries we need to read the data

```
data-pd.read_csv("used_cars_data.csv")

data.head()
```

Analyzing the data:

a) Check for duplicates

nunique() based on several unique values in each column and the data description, we can identify the continuous and categorical columns in the data. Duplicated data can be handled or removed based on further analysis

```
data.nunique()
```

b) Missing Values Calculation

isnull() is widely been in all pre-processing steps to identify null values in the data

```
data.isnull().sum()
```

The following code helps to calculate the percentage of missing values in each column

```
Data.isnull().sum()/(len(data))*
```

Data reduction

Some columns or variables can be dropped if they do not add value to our analysis.

In our dataset, the column S.No have only ID values, assuming they don't have any predictive power to predict the dependent variable

```
# remove s.no column from data

Data=data.drop(['s.no'],axis=1)
```

Creating Features

If we see the sample data, the column "Year" shows the manufacturing year of the car.

It would be difficult to find the car's age if it is in year format as the Age of the car is a contributing factor to Car Price.

Introducing a new column, "Car_Age" to know the age of the car

```
From datetime import date

Date.today().year

Data['car_age']=date.today().year_data['year']

Date.head()
```

Let's split the name and introduce new variables "Brand" and "Model"

```
Date['brand']=data.name.str.split().str.get(0)

Data['model']=data.name.str.split().str.get(1)+data.name.str.split().str.get(2)

Data['name','brand','model']
```

EDA Analysis

Separate Numerical and categorical variables for easy analysis

```
[39] cat_cols=data.select_dtypes(include=['object']).columns
     num_cols = data.select_dtypes(include=np.number).columns.tolist()
     print("Categorical Variables:")
     print(cat_cols)
     print("Numerical Variables:")
     print(num_cols)
```

```
Categorical Variables:
Index(['Name', 'Location', 'Fuel_Type', 'Transmission', 'Owner_Type',
       'Mileage', 'Engine', 'Power', 'New_Price', 'Brand', 'Model'],
      dtype='object')
Numerical Variables:
['Year', 'Kilometers_Driven', 'Seats', 'Price', 'Car_Age']
```

Exploring your data with basic statistical functions is a crucial first step in any data analysis or machine learning project. These functions help you understand the distribution, central tendency, and variability of your data. Here are some key statistical functions and techniques used in data exploration:

1. Descriptive Statistics: Mean

   The average of a dataset.

   import numpy as np data = [1, 2, 3, 4, 5]

   mean = np.mean(data)

   print(f'Mean: {mean}')

   Median

   The middle value of a dataset.

   median = np.median(data)

   print(f'Median: {median}')

   Mode

   The most frequent value(s) in a dataset.

   from scipy import stats mode = stats.mode(data)

```
print(f'Mode: {mode.mode[0]}')
```

Standard Deviation

Measures the amount of variation or dispersion in a dataset.

```
python
std_dev = np.std(data)
print(f'Standard Deviation: {std_dev}')
```

Variance

The average of the squared differences from the mean.

```
python
variance = np.var(data)
print(f'Variance: {variance}')
```

2. Data Distribution

Min and Max

The minimum and maximum values in a dataset.

```
python
min_value = np.min(data)
max_value = np.max(data)
print(f'Min: {min_value}, Max: {max_value}')
```

Quartiles and Interquartile Range (IQR)

Quartiles divide the data into four equal parts, and IQR measures the middle 50% of the data.

```
python
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
print(f'Q1: {q1}, Q3: {q3}, IQR: {iqr}')
```

3. Correlation and Covariance

Correlation

Measures the strength and direction of the relationship between two variables.

```
python
data2 = [5, 4, 3, 2, 1]

correlation = np.corrcoef(data, data2)[0, 1]
print(f'Correlation: {correlation}')
```

Covariance

Indicates the direction of the linear relationship between variables.

```
python
covariance = np.cov(data, data2)[0, 1]
print(f'Covariance: {covariance}')
```

Data Visualization

Visualizing your data helps in understanding its distribution and identifying patterns or outliers.

Histogram

```
python
import matplotlib.pyplot as plt

plt.hist(data, bins=5, edgecolor='black')
plt.title('Histogram')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

Box Plot

```
python
plt.boxplot(data)
plt.title('Box Plot')
plt.ylabel('Value')
plt.show()
```

Scatter Plot

```
python
plt.scatter(data, data2)
plt.title('Scatter Plot')
plt.xlabel('Data 1')
plt.ylabel('Data 2')
plt.show()
```

5. Pandas Descriptive Functions

Using pandas, you can easily get a summary of your data.

```
import pandas as pd
df = pd.DataFrame({'data': data, 'data2': data2})
print(df.describe())
```