IDS ASSIGNMENT

Tejas Pebam
BCA (A)
Registration No. :- 2411021240018
GitHub link:- TejasPebam/IDS-ASSIGNMENT-

What is Data Science?

Data Science is a field that combines mathematics, statistics, computer science, and domain-specific knowledge to interpret and analyze data. It focuses on asking the right questions and using available data to find the answers. Essentially, data science involves turning raw data into valuable insights by cleaning, analyzing, and then using those findings to make informed decisions or forecast future events.

In simple terms, data science helps people and businesses make better decisions, solve problems, or predict future trends by utilizing data. It is widely applied across industries like healthcare, business, and technology to improve efficiency and provide better services.

Key Components of Data Science:

1. **Data Collection:** The first step is gathering data from a variety of sources, such as sensors, surveys, websites, and more.
2. **Data Processing:** Once the data is collected, it needs to be cleaned and formatted, addressing any errors or missing values to make it ready for analysis.
3. **Data Analysis:** After cleaning, data scientists use various techniques—such as statistical methods or algorithms—to analyze the data and uncover patterns or insights.
4. **Visualization:** The results of the analysis are then presented in a clear and understandable way, often using charts, graphs, or dashboards.
5. **Decision-Making:** The ultimate goal of data science is to use the insights to guide decision-making, whether it's recommending a product, solving a problem, or predicting trends.

The Data Science Process: CRISP-DM

CRISP-DM (Cross-Industry Standard Process for Data Mining) is a widely used framework in data science that helps guide projects through a structured approach. It consists of six phases:

1. **Business Understanding:** The first step is to understand the problem you're aiming to solve. You need to identify the goals of the business and what success would look like.
2. **Data Understanding:** This phase involves exploring the data to understand its structure, quality, and potential issues such as missing values.
3. **Data Preparation:** Once you understand the data, you prepare it for analysis. This may involve cleaning, reformatting, or removing irrelevant information.

4. **Modeling:** In this step, you apply various algorithms or models to the data. These could range from machine learning models to simpler techniques, depending on the problem.
5. **Evaluation:** After creating the model, it's crucial to evaluate its effectiveness. This may involve testing it with new data or using performance metrics to assess how well it addresses the problem.
6. **Deployment:** Finally, once the model is proven effective, it is deployed in real-world applications, such as integrating it into a website or system for automatic use.

2) Case Study question

The primary business goal of the Netflix Recommendation System is to boost user engagement by offering highly personalized recommendations for movies and TV shows tailored to each user's preferences. By analyzing user behavior, tastes, and viewing history, Netflix strives to provide a smooth and enjoyable content discovery experience. This level of personalization not only enhances user satisfaction by suggesting content they're likely to enjoy, but also encourages them to spend more time on the platform, increasing overall engagement. Additionally, an effective recommendation system helps retain users by making them feel valued, thereby reducing the chance of subscription cancellations. It also helps minimize decision fatigue by simplifying the content selection process, making it easier for users to find relevant and appealing options without wading through an overwhelming number of titles. In the end, the recommendation system is essential in driving user satisfaction, retention, and usage, all of which contribute to Netflix's long-term success and its competitive advantage in the streaming market.

Code Block

[1]:
```
import pandas as pd
a=pd.read_csv(r'C:\Users\tejas\Downloads\students.csv')
b=pd.read_csv(r'C:\Users\tejas\Downloads\details.csv')
```

[2]:
```
a
```

[2]:

	StudentID	Name	Marks
0	101	Alice	85
1	102	Bob	90
2	103	Charlie	88
3	104	David	92

[3]:
```
b
```

[3]:

	StudentID	Age	Grade
0	101	20	A
1	102	21	B
2	103	22	A
3	105	19	C

[4]:
```
c=pd.merge(a,b,on='StudentID',how='inner')
c
#Inner join prints both the data set which is common
```

[4]:

	StudentID	Name	Marks	Age	Grade
0	101	Alice	85	20	A
1	102	Bob	90	21	B
2	103	Charlie	88	22	A

[5]:
```
c=pd.merge(a,b,on='StudentID',how='right')
c
#Right outer join prints all the right values and the corresponding values and those values which are not corresponding will be printed as NaN
```

[5]:

	StudentID	Name	Marks	Age	Grade
0	101	Alice	85.0	20	A
1	102	Bob	90.0	21	B
2	103	Charlie	88.0	22	A
3	105	NaN	NaN	19	C

[6]:
```
c=pd.merge(a,b,on='StudentID',how='left')
c
#Left outer join prints all the left values and the corresponding values and those values which are not corresponding will be printed as NaN
```

[6]:

	StudentID	Name	Marks	Age	Grade
0	101	Alice	85	20.0	A
1	102	Bob	90	21.0	B
2	103	Charlie	88	22.0	A
3	104	David	92	NaN	NaN

[7]:
```
c=pd.merge(a,b,on='StudentID',how='outer')
c
#Outer join prints the all values right and left.
#outer join prints the corresponding values from both data sets as it is and those who don't correspond will be printed as NaN
```

[7]:

	StudentID	Name	Marks	Age	Grade
0	101	Alice	85.0	20.0	A
1	102	Bob	90.0	21.0	B
2	103	Charlie	88.0	22.0	A
3	104	David	92.0	NaN	NaN
4	105	NaN	NaN	19.0	C

[8]:
```
d=c.set_index('StudentID')
d
```

[8]:

	Name	Marks	Age	Grade
StudentID				
101	Alice	85.0	20.0	A
102	Bob	90.0	21.0	B
103	Charlie	88.0	22.0	A
104	David	92.0	NaN	NaN
105	NaN	NaN	19.0	C

[9]:
```
d.reset_index(inplace=True)
d
```

[9]:

	StudentID	Name	Marks	Age	Grade
0	101	Alice	85.0	20.0	A
1	102	Bob	90.0	21.0	B
2	103	Charlie	88.0	22.0	A
3	104	David	92.0	NaN	NaN
4	105	NaN	NaN	19.0	C

10]:
```
d.to_csv('merged_students_details.csv')
d
```

10]:

	StudentID	Name	Marks	Age	Grade
0	101	Alice	85.0	20.0	A
1	102	Bob	90.0	21.0	B
2	103	Charlie	88.0	22.0	A
3	104	David	92.0	NaN	NaN
4	105	NaN	NaN	19.0	C

```
e=pd.read_csv(r'C:\Users\tejas\Downloads\diabetes.csv')
e
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

```
e.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
print(e.shape)
```

```
(768, 9)
```

```
e.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
e.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
e['BMI'].median()
```

```
32.0
```

```
e['Glucose'].median()
```

```
117.0
```

[18]:
```
e['BMI']=e['BMI'].replace(0,32.0)
e
```

[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

[19]:
```
e['Glucose']=e['Glucose'].replace(0,117.0)
e
```

[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns