INTRODUCTION TO DATA SCIENCE

NAME: Moideen Nihal
CLASS: BCA-A
REGISTER ID: 2411021240057
GITHUB: https://github.com/mo-nihal

Part 1: Theoretical Understanding

1. Define Data Science:

o What is Data Science? Discuss its key components and the CRISP-DM process.
o Explain how the CRISP-DM framework is applied in solving real-world problems
(e.g., predicting customer churn or recommending movies).

2. Case Study Questions:

- From the case studies in the "Module 1 Case Studies" file, answer the following:
- ▪ What is the main business objective of the Netflix Recommendation System?

1.What is Data Science?

Data science is a field that combines math, statistics, computer science, and subject-specific knowledge to understand and analyze data. Essentially, it involves asking the right questions and using the available data to find the answers. It's about transforming raw data into meaningful insights by cleaning, analyzing, and leveraging the findings to make informed decisions or predict future outcomes. In simpler terms, data science helps individuals and businesses use data to solve problems, make decisions, or predict potential events. It's widely applied across various industries today, such as healthcare, business, and technology, to improve efficiency and create better services.

Key Components of Data Science:

1. **Data Collection**: The first step is gathering data from different sources. This data could come from sensors, surveys, websites, or anything else that collects information.
2. **Data Processing**: Once data is collected, it's usually not in a perfect form. You'll need to clean it up. This means fixing errors, dealing with missing values, and formatting the data so it's ready for analysis.
3. **Data Analysis**: After cleaning up the data, data scientists use different techniques to figure out what the data is telling them. This could be through statistical methods or using algorithms to find patterns in the data.
4. **Visualization**: Once you have your results, you need to present them in a way that makes sense. This could be through graphs, charts, or dashboards that help people easily understand the insights you've found.
5. **Decision-Making**: The ultimate goal of data science is to use the insights you've gathered to make decisions. Whether it's recommending a product, solving a problem, or predicting a trend, data science helps people make better choices.

Data Science Process: CRISP-DM

The **CRISP-DM** method is one of the most common approaches used in data science. It stands for Cross-Industry Standard Process for Data Mining. This method helps guide data scientists through their projects in a structured way. There are six stages in CRISP-DM:

1. **Business Understanding**: This step is all about understanding the problem you're trying to solve. You need to know what the business wants to achieve and what success would look like.
2. **Data Understanding**: In this stage, you start exploring the data. You check what kind of data you have, where it's coming from, and what problems might be in the data, like missing values.

3. **Data Preparation**: After understanding the data, you get it ready for analysis. This might involve cleaning the data, changing the format, or removing irrelevant parts.
4. **Modeling**: In this step, you apply different algorithms or models to the data. This could be a machine learning model or something simpler, depending on what the problem is.
5. **Evaluation**: Once the model is ready, you need to check if it actually solves the problem. This could involve testing the model with new data or using specific metrics to see how well it's performing.
6. **Deployment**: Finally, once the model is working well, you implement it in the real world. This could mean integrating it into a website or a system that automatically uses the model's predictions.

1.*Case Study: Netflix Recommendation System*

Problem: Netflix wants to suggest movies and TV shows to users based on what they have watched before. But how do they know what each user will like?
Dataset: Netflix uses datasets like the **MovieLens Dataset** to train its recommendation system. This dataset includes information about movies, ratings, and users.
Some of the columns in the dataset include:

- **User ID**: The unique ID for each user.
- **Movie ID**: The unique ID for each movie.
- **Rating**: The rating a user gives to a movie, typically from 1 to 5 stars.
- **Timestamp**: When the rating was given.
- **Movie Metadata**: This includes information about the movie, such as title, genre, and release year.

Applying the Data Science Process to Netflix:

1. **Business Understanding**:
    - **Goal**: The main goal is to suggest content to users that they'll enjoy, which helps keep them engaged on the platform.

- **Impact**: If Netflix can give good recommendations, users will be more likely to stay subscribed and watch more content.

2. **Data Understanding**:
   - **Exploring the Data**: First, we check how many users, movies, and ratings are in the dataset. This helps us understand how big the data is and what type of data we're working with.
   - **Rating Distribution**: It's also useful to see how ratings are spread out—are most ratings high or low? This gives us an idea of how users generally interact with the platform.
3. **Data Preparation**:
   - **Cleaning Data**: Sometimes, movie metadata might be missing, like the genre or year of release. These gaps need to be fixed before the data can be used.
   - **Timestamp Transformation**: The timestamp is a bit tricky, so we might convert it into something more readable, like the year or month, to see trends over time.
   - **Encoding Genres**: Movies have genres like Action, Drama, or Comedy. We need to turn these categories into numbers so the model can understand them.
4. **Modeling**:
   - **Collaborative Filtering**: Netflix uses collaborative filtering, which means it finds patterns in users' ratings. If users like similar things, the system suggests what else those users liked.
   - **Singular Value Decomposition (SVD)**: SVD is a technique that breaks down the large matrix of ratings into smaller parts, which makes it easier to predict ratings for movies users haven't seen yet.
   - **Content-Based Filtering**: Another method is content-based filtering, where the system suggests movies based on their features, like genre or the actors in them.
5. **Evaluation**:
   - **Testing the Model**: To make sure the model works, we split the data into training and testing sets. This way, we can see how well the model does on new data it hasn't seen before.

- **Metrics**: We check how accurate the model is by using something like **Root Mean Square Error (RMSE)**. The lower the RMSE, the better the model.

6. **Deployment**:
   - **Integrating the Model**: Once the model works well, it's put into Netflix's recommendation engine, so users start getting suggestions based on their tastes.
   - **Real-Time Recommendations**: The system keeps improving as users watch more content, constantly learning from new ratings and keeping recommendations fresh.

Conclusion

Data science is an exciting field because it provides solutions to real-world problems through data analysis. The CRISP-DM framework ensures that data science projects are executed in a systematic and efficient manner, from problem definition to solution deployment.

A great example of data science in action is Netflix's recommendation system. By analyzing user viewing habits and preferences, Netflix is able to suggest movies and shows that align with individual tastes, keeping users engaged and fostering loyalty to the platform.

Through techniques such as collaborative filtering, content-based filtering, and SVD, Netflix guarantees that users receive up-to-date and relevant recommendations, enhancing the overall experience on the platform.

CODE BLOCK

[37]:
```
import pandas as pd
a=pd.read_csv(r"C:\Users\hp\Desktop\students.csv")
b=pd.read_csv(r"C:\Users\hp\Desktop\details.csv")
```

[39]:
```
a
```

[39]:

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

[41]:
```
b
```

[41]:

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

[43]:
```
c=pd.merge(a,b,on='StudentID',how='inner')
c
#Inner join prints both the data set which is common
```

[43]:

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

[45]:
```
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
```

[45]:

	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

[47]:
```
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
```

[47]:

	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

[49]:
```
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
```

[49]:

	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

[51]:
```
d=c.set_index('StudentID')
d
```

[51]:

StudentID	Name	Marks	Age	Grade
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

[53]:
```
d.reset_index(inplace=True)
d
```

[53]:

	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

[55]:
```
d.to_csv('merged_students_details.csv')
d
```

[55]:

	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

[59]:
```
e=pd.read_csv(r"C:\Users\hp\Desktop\diabetes.csv")
e
```

[59]:

	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

[61]:
```
print(e.shape)
```

```
(768, 9)
```

[63]:
```
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
```

[65]:
```
e.describe()
```

[65]:

	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

[67]:
```
e['BMI'].median()
```

[67]:
```
32.0
```

[69]:
```
e['Glucose'].median()
```

[69]:
```
117.0
```

[71]:
```
e['BMI']=e['BMI'].replace(0,32.0)
e
```

[71]:

	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

[73]:
```
e['Glucose']=e['Glucose'].replace(0,e['Glucose'].median())
e
```

[73]:

	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