

Introduction to Data Science Assignment 1

Name: Shashini Shanmugan
Course: BCA(Sec A)
Registration Number - 2411021240013
Github Repository link -

Part 1: Theoretical Understanding

1. Define Data Science

Q. What is Data Science? Discuss its key components and the CRISP-DM process.

Data Science is an interdisciplinary field that uses mathematics, statistics, computer science, and domain knowledge to extract meaningful insights from data.

Key Components:

- Data Collection: Gathering raw data from various sources.
- Data Processing: Cleaning and organizing the data.
- Data Analysis: Using statistical and computational methods to derive insights.
- Visualization: Presenting results in an understandable format (e.g., graphs, dashboards).
- Decision-Making: Applying insights to solve real-world problems.

CRISP-DM process:

1. Business Understanding: Define objectives and questions.
2. Data Understanding: Explore the data to understand its structure.
3. Data Preparation: Clean, transform, and organize the data.
4. Modeling: Apply algorithms to analyze and predict.
5. Evaluation: Check if the model answers the problem.
6. Deployment: Implement the solution in real-world scenarios.

Q.Explain how the CRISP-DM framework is applied in solving real-world problems (e.g., predicting customer churn or recommending movies).

Predicting Customer Churn in Telecom

Problem Statement How can a telecom company predict which customers are likely to stop using their services?

Dataset

Source: Telco Customer Churn[<https://www.kaggle.com/datasets/blastchar/telcocustomer-churn>]

Columns:

Customer ID

Demographics: Gender, age group, etc.

Service details: Internet service, contract type, monthly charges.

Churn (Yes/No).

CRISP DM Process

1. Business Understanding:

Goal: Reduce churn rate by identifying at-risk customers.

Impact: Increase revenue by targeting retention efforts.

2. Data Understanding:

Analyze churn rate across demographics.

Understand correlations between service features and churn.

3. Data Preparation:

Handle missing values (e.g., impute missing charges).

Convert categorical variables (e.g., gender, contract type) to numerical data.

Scale numerical features like monthly charges.

4. Modeling:

Train a classification model (e.g., Logistic Regression, Random Forest) to predict churn.

Use features like contract type, monthly charges, and tenure.

5. Evaluation:

Use metrics like Accuracy, Precision, Recall, and F1 Score.

Evaluate the model on a confusion matrix to understand false positives and negatives.

6. Deployment:

Provide alerts for high-risk customers to customer service teams.

Implement proactive offers and discounts to retain customers.

Netflix Recommendation System

Problem Statement: How can Netflix recommend personalized movies or TV shows to users based on their preferences?

Dataset

Source: MovieLens Dataset (Free public dataset). [<https://grouplens.org/datasets/movielens/>]

Columns:

- User ID
- Movie ID
- Rating (1-5)
- Timestamp
- Movie metadata (title, genres, release year).

CRISP-DM Process-

1. Business Understanding:

Goal: Improve user engagement by suggesting content they're likely to enjoy.

Impact: Increased user satisfaction and retention.

2. Data Understanding:

Explore the dataset: Number of users, movies, and ratings.

Analyze distribution of ratings and popular genres.

3. Data Preparation:

Handle missing data in movie metadata.

Transform timestamp into human-readable format.

One-hot encode genres for analysis.

4. Modeling:

Use Collaborative Filtering to predict user preferences:

Find similar users and suggest movies they liked.

Train a recommendation model (e.g., Singular Value Decomposition).

Alternative: Content-based filtering using movie metadata.

5. Evaluation:

Split data into training and test sets.

Use metrics like Root Mean Square Error (RMSE) to measure model accuracy.

6. Deployment:

Integrate the model into a recommendation engine.

Provide real-time recommendations on the Netflix platform.

2. Case Study Questions:

From the case studies in the "Module 1 Case Studies" file, answer the following:

Q.What is the main business objective of the Netflix Recommendation System?

The main business objective of the Netflix Recommendation System is to improve user engagement by suggesting content they're likely to enjoy. This will increase the user satisfaction and retention.

Part 2: Data Manipulation and Joins Use the following two CSV files for practical tasks:

File 1: students.csv:

StudentID,Name,Marks

101,Alice,85

102,Bob,90

103,Charlie,88

104,David,92

File 2: details.csv:

StudentID,Age,Grade

101,20,A

102,21,B

103,22,A

105,19,C

Tasks:

1. Load the Datasets:

Load students.csv and details.csv into pandas DataFrames.

2. Perform Joins:

Merge the two DataFrames using the following join types:

Inner Join

Left Join

Right Join

Outer Join

For each join type, describe the difference in the results.

3. Set Index:

Set StudentID as the index for the merged DataFrame.

Reset the index back to default.

4. Save the Results: Save the result of the outer join to a new CSV file called merged_students_details.csv.

```
[1] import pandas as pd
```

```
[2] students = pd.read_csv(r"/content/students.csv")
```

```
[3] students
```

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

Next steps: [Generate code with students](#) [View recommended plots](#) [New interactive sheet](#)

```
[4] details = pd.read_csv(r"/content/details.csv")
```

```
details
```

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

Next steps: [Generate code with details](#) [View recommended plots](#) [New interactive sheet](#)

```
[10] #Inner Join
merged_df=pd.merge(students,details,on="Student ID",how="inner")
merged_df
```

	Student ID	Name	Marks	Age	Grade
--	------------	------	-------	-----	-------

✓ 0s completed at 10:45AM

```
[10] #Inner Join
merged_df=pd.merge(students,details,on="Student ID",how="inner")
merged_df
```

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

Next steps: [Generate code with merged_df](#) [View recommended plots](#) [New interactive sheet](#)

```
#Left Join
merged_df1=pd.merge(students,details,on="Student ID",how="left")
merged_df1
```

	Student ID	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

Next steps: [Generate code with merged_df1](#) [View recommended plots](#) [New interactive sheet](#)

```
[12] #Right Join
merged_df2=pd.merge(students,details,on="Student ID",how="right")
merged_df2
```

	Student ID	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

✓ 0s completed at 10:45 AM

```
[13] #Outer Join
merged_df3=pd.merge(students, details, on="Student ID", how="outer")
merged_df3
```

	Student ID	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

Next steps: [Generate code with merged_df3](#) [View recommended plots](#) [New interactive sheet](#)

```
merged_df3.set_index("Student ID", inplace=True)
merged_df3
```

	Student ID	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

Next steps: [Generate code with merged_df3](#) [View recommended plots](#) [New interactive sheet](#)

```
[15] merged_df3.to_csv("merged_students_details.csv" , index=False)
```

```
[19] merged_students_details=pd.DataFrame(merged_df3)
```

```
merged_students_details
```

✓ 0s completed at 10:46 AM

```
[15] merged_df3.to_csv("merged_students_details.csv" , index=False)
```

```
[19] merged_students_details=pd.DataFrame(merged_df3)
```

merged_students_details

	Name	Marks	Age	Grade
Student ID				
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

Next steps: [Generate code with merged_students_details](#) [View recommended plots](#) [New interactive sheet](#)

0s completed at 10:46 AM

Part 3 - Exploratory Data Analysis (EDA)

Dataset: PIMA Indians Diabetes Dataset (use the dataset in the attached case studies or download it from Kaggle).

Tasks:

1. Load the Dataset:

Load the dataset into a pandas DataFrame.

Display its first five rows, shape, and basic statistics.

2. Handle Missing Values:

Identify and replace zeros in columns like Glucose or BMI with the median value of the respective column.

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell imports pandas and reads a CSV file. The second cell displays the first five rows of the dataset using `df.head()`.

```
import pandas as pd

[4] df=pd.read_csv(r"/content/diabetes.csv")
df
```

	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 x 9 columns

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

```
[5] df.head()#by default prints the first 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

0s completed at 11:06 AM

0s

df.head()#by default prints the first 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

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

0s

[6] df.tail()#by default prints the last 5

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

0s

[7] print(df.shape)

(768, 9)

0s

[8] #Handle Missing Values:
df.fillna(value="100")
df

	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

0s

completed at 11:06 AM

0s

#Handle Missing Values:
df.fillna(value="100")
df

	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 x 9 columns

Next steps:

Generate code with df

View recommended plots

New interactive sheet

0s

[9] df

	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

0s

completed at 11:06 AM

df

	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

Next steps:

Generate code with df

View recommended plots

New interactive sheet

0s

[10] # Replace zeros with the median in specific columns

df

df["Glucose"] = df["Glucose"].replace(0, df["Glucose"].median())

df["BMI"] = df["BMI"].replace(0, df["BMI"].median())

0s

[11] df

	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
...

0s

completed at 11:06 AM

1

Age Group	Percentage
18-29	~65%
30-49	~55%
50-69	~45%
70+	~35%