

EXPERIMENT NO: 1

Aim: Introduction to Data science and Data preparation using Pandas steps.

Theory:

Data preparation is a fundamental step in data science, involving the cleaning and transformation of raw data into a structured and analyzable format. Pandas, a powerful Python library, provides efficient tools for handling missing values, encoding categorical data, and scaling numerical features. Proper preprocessing enhances dataset quality, ensuring consistency and reliability for further analysis and machine learning models.

Problem Statement:

The Placement Data dataset contains various attributes related to students' academic performance, placement status, and salary packages. The objective of this experiment is to:

- Identify key trends in student placements based on academic performance.
- Analyze the distribution of salary packages.
- Handle missing data and remove inconsistencies.
- Standardize and normalize the data for further analysis.

By cleaning the placement dataset and applying data preprocessing steps, the goal is to improve data reliability, analyze student performance trends, and provide valuable insights for academic and recruitment decisions.

Dataset Overview:

The dataset provides detailed information about student placements, academic performance, and salary distributions. It contains multiple columns, each capturing specific attributes related to students' educational backgrounds and employment outcomes. Below is a breakdown of the columns and their relevance: The dataset provides insights into student placements, containing columns such as:

1. **StudentID:** Unique identifier for each student.
2. **CGPA:** Cumulative Grade Point Average of the student.
3. **Internships:** Number of internships completed.
4. **Projects:** Number of academic or industry projects undertaken.
5. **Workshops/Certifications:** Number of workshops attended or certifications earned.
6. **Aptitude Test Score:** Score obtained in the aptitude test.
7. **Soft Skills Rating:** Rating of soft skills on a predefined scale.
8. **Extracurricular Activities:** Participation in extracurricular activities.
9. **Placement Training:** Whether the student underwent placement training (Yes/No).
10. **SSC Marks:** Secondary school exam scores.
11. **HSC Marks:** Higher secondary exam scores.

12. Placement Status: Whether the student was placed or not.

Steps:

Loading The Dataset

```
✓ 2s [1] import pandas as pd
```

```
✓ 0s df = pd.read_csv('Data.csv')
```

Description of the dataset

a. Information about dataset

```
✓ 0s df.info()
```


```
↗ <class 'pandas.core.frame.DataFrame'>  
RangeIndex: 10000 entries, 0 to 9999  
Data columns (total 12 columns):  
#   Column                                Non-Null Count  Dtype  
---  ---                                -  
0   StudentID                            10000 non-null  int64  
1   CGPA                                  10000 non-null  float64  
2   Internships                          10000 non-null  int64  
3   Projects                             10000 non-null  int64  
4   Workshops/Certifications             10000 non-null  int64  
5   AptitudeTestScore                   10000 non-null  int64  
6   SoftSkillsRating                     10000 non-null  float64  
7   ExtracurricularActivities            10000 non-null  object  
8   PlacementTraining                    10000 non-null  object  
9   SSC_Marks                            10000 non-null  int64  
10  HSC_Marks                            10000 non-null  int64  
11  PlacementStatus                      10000 non-null  object  
dtypes: float64(2), int64(7), object(3)  
memory usage: 937.6+ KB
```

b. Description of Dataset

```
# Display basic information about the dataset
print("Dataset Information:")
print(df.info())

# Display summary statistics
print("\nDataset Description:")
print(df.describe())

# Display the first few rows of the dataset
print("\nFirst 5 Rows of the Dataset:")
print(df.head())
```

 Dataset Information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   StudentID                            10000 non-null  int64
1   CGPA                                 10000 non-null  float64
2   Internships                          10000 non-null  int64
3   Projects                             10000 non-null  int64
4   Workshops/Certifications             10000 non-null  int64
5   AptitudeTestScore                   10000 non-null  int64
6   SoftSkillsRating                    10000 non-null  float64
7   ExtracurricularActivities           10000 non-null  object
8   PlacementTraining                   10000 non-null  object
9   SSC_Marks                           10000 non-null  int64
10  HSC_Marks                           10000 non-null  int64
11  PlacementStatus                     10000 non-null  object
dtypes: float64(2), int64(7), object(3)
memory usage: 937.6+ KB
None
```

	Workshops/Certifications	AptitudeTestScore	SoftSkillsRating
count	10000.000000	10000.000000	10000.000000
mean	1.013200	79.449900	4.323960
std	0.904272	8.159997	0.411622
min	0.000000	60.000000	3.000000
25%	0.000000	73.000000	4.000000
50%	1.000000	80.000000	4.400000
75%	2.000000	87.000000	4.700000
max	3.000000	90.000000	4.800000

	SSC_Marks	HSC_Marks
count	10000.000000	10000.000000
mean	69.159400	74.501500
std	10.430459	8.919527
min	55.000000	57.000000
25%	59.000000	67.000000
50%	70.000000	73.000000
75%	78.000000	83.000000
max	90.000000	88.000000

Dataset Description:

	StudentID	CGPA	Internships	Projects
count	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.500000	7.698010	1.049200	2.026600
std	2886.89568	0.640131	0.665901	0.867968
min	1.000000	6.500000	0.000000	0.000000
25%	2500.750000	7.400000	1.000000	1.000000
50%	5000.500000	7.700000	1.000000	2.000000
75%	7500.250000	8.200000	1.000000	3.000000
max	10000.000000	9.100000	2.000000	3.000000

First 5 Rows of the Dataset:

	StudentID	CGPA	Internships	Projects	Workshops/Certifications
0	1	7.5	1	1	1
1	2	8.9	0	3	2
2	3	7.3	1	2	2
3	4	7.5	1	1	2
4	5	8.3	1	2	2

	AptitudeTestScore	SoftSkillsRating	ExtracurricularActivities
0	65	4.4	No
1	90	4.0	Yes
2	82	4.8	Yes
3	85	4.4	Yes
4	86	4.5	Yes

	PlacementTraining	SSC_Marks	HSC_Marks	PlacementStatus
0	No	61	79	NotPlaced
1	Yes	78	82	Placed
2	No	79	80	NotPlaced
3	Yes	81	80	Placed
4	Yes	74	88	Placed

Drop columns that aren't useful.



```
columns_to_drop = ['StudentID']
df.drop(columns=columns_to_drop, inplace=True)
print(df.describe)
```

<bound	method	NDFrame.describe	of	CGPA	Internships	Projects	Workshops/Certifications	\
0	7.5	1	1			1		
1	8.9	0	3			2		
2	7.3	1	2			2		
3	7.5	1	1			2		
4	8.3	1	2			2		
...		
9995	7.5	1	1			2		
9996	7.4	0	1			0		
9997	8.4	1	3			0		
9998	8.9	0	3			2		
9999	8.4	0	1			1		

	AptitudeTestScore	SoftSkillsRating	ExtracurricularActivities	\
0	65	4.4		No
1	90	4.0		Yes
2	82	4.8		Yes
3	85	4.4		Yes
4	86	4.5		Yes
...
9995	72	3.9		Yes
9996	90	4.8		No
9997	70	4.8		Yes
9998	87	4.8		Yes
9999	66	3.8		No

	PlacementTraining	SSC_Marks	HSC_Marks	PlacementStatus
0	No	61	79	NotPlaced
1	Yes	78	82	Placed
2	No	79	80	NotPlaced
3	Yes	81	80	Placed
4	Yes	74	88	Placed
...
9995	No	85	66	NotPlaced
9996	No	84	67	Placed
9997	Yes	79	81	Placed
9998	Yes	71	85	Placed
9999	No	62	66	NotPlaced

[10000 rows x 11 columns]>

Thus the columns now present in dataset are:

```

print("Dataset Information:")
print(df.info())

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 11 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   CGPA                                  10000 non-null  float64
1   Internships                          10000 non-null  int64
2   Projects                             10000 non-null  int64
3   Workshops/Certifications              10000 non-null  int64
4   AptitudeTestScore                    10000 non-null  int64
5   SoftSkillsRating                     10000 non-null  float64
6   ExtracurricularActivities            10000 non-null  object
7   PlacementTraining                    10000 non-null  object
8   SSC_Marks                            10000 non-null  int64
9   HSC_Marks                            10000 non-null  int64
10  PlacementStatus                      10000 non-null  object
dtypes: float64(2), int64(6), object(3)
memory usage: 859.5+ KB
None

```

Take care of missing data.

c. Drop rows with maximum missing values.

```
▶ print(f"Dataset Shape before Dropping Rows: {df.shape}")
# Drop rows with the highest number of missing values
threshold = len(df.columns) * 0.5 # Drop rows where over 50% of columns are missing
df = df.dropna(thresh=threshold)

print(f"Dataset Shape After Dropping Rows: {df.shape}")
```

```
↗ Dataset Shape before Dropping Rows: (28671, 10)
Dataset Shape After Dropping Rows: (28671, 10)
```

```
[15] print(df.isnull().sum())
```

```
↗ CGPA 0
Internships 0
Projects 0
Workshops/Certifications 0
AptitudeTestScore 0
SoftSkillsRating 0
ExtracurricularActivities 0
PlacementTraining 0
SSC_Marks 0
HSC_Marks 0
PlacementStatus 0
dtype: int64
```

The output of `print(df.isnull().sum())` indicates that there are no missing values in any of the columns of the dataset. Each column has a count of 0, meaning that all records have valid data for every feature.

Data Preprocessing

- The code uses `dropna()` to remove missing values.
- `fillna()` is used to fill missing values with the mean, specifically for numerical columns.
- `get_dummies()` is applied to categorical variables like Placement Status, Placement Training, and Extracurricular Activities, with `drop_first=True` to avoid multicollinearity.

```
[8] df.dropna(thresh=df.shape[1] - 1, inplace=True)
```

```
df.fillna(df.select_dtypes(include=['number']).mean(), inplace=True)
```

```
df_encoded = pd.get_dummies(df, columns=['PlacementStatus', 'PlacementTraining', 'ExtracurricularActivities'], drop_first=True)
print(df_encoded)
```

	CGPA	Internships	Projects	Workshops/Certifications	\
0	7.5	1	1		1
1	8.9	0	3		2
2	7.3	1	2		2
3	7.5	1	1		2
4	8.3	1	2		2
...
9995	7.5	1	1		2
9996	7.4	0	1		0
9997	8.4	1	3		0
9998	8.9	0	3		2
9999	8.4	0	1		1

	AptitudeTestScore	SoftSkillsRating	SSC_Marks	HSC_Marks	\
0		65	4.4	61	79
1		90	4.0	78	82
2		82	4.8	79	80
3		85	4.4	81	80
4		86	4.5	74	88
...	
9995		72	3.9	85	66
9996		90	4.8	84	67
9997		70	4.8	79	81
9998		87	4.8	71	85
9999		66	3.8	62	66

	PlacementStatus_Placed	PlacementTraining_Yes	\
0	False		False
1	True		True
2	False		False
3	True		True
4	True		True
...
9995	False		False
9996	True		False
9997	True		True
9998	True		True
9999	False		False

	ExtracurricularActivities_Yes
0	False
1	True
2	True
3	True
4	True
...	...
9995	True
9996	False
9997	True
9998	True
9999	False

[10000 rows x 11 columns]

Dataset Features

- The dataset includes Aptitude Test Scores, Soft Skills Ratings, SSC Marks, HSC Marks, Internships, Projects, and Workshops/Certifications.
- It tracks whether a student is placed or not.

Saving Processed Data

- The cleaned and encoded dataset is saved as "new_data.csv".

```
[ ] df_encoded.to_csv("new_data.csv", index = False)

[ ] df_numeric = df.select_dtypes(include=[float, int])

Q1 = df_numeric.quantile(0.25)
Q3 = df_numeric.quantile(0.75)
IQR = Q3 - Q1

outliers = (df_numeric < (Q1 - 1.5 * IQR)) | (df_numeric > (Q3 + 1.5 * IQR))

df_outliers_marked = df.copy()

for col in df_numeric.columns:
    df_outliers_marked[col] = df[col].where(~outliers[col], other="OUTLIER")

print(df_outliers_marked)
```

	HSC_Marks	PlacementStatus
0	79	NotPlaced
1	82	Placed
2	80	NotPlaced
3	80	Placed
4	88	Placed
...
9995	66	NotPlaced
9996	67	Placed
9997	81	Placed
9998	85	Placed
9999	66	NotPlaced

[10000 rows x 11 columns]

Placement Analysis

- There are binary indicators for Placement Status, Placement Training participation, and Extracurricular Activities.
- The dataset seems to have 10,000 rows and 11 columns.

Outlier Detection

- Uses Interquartile Range (IQR) method to detect outliers in numerical data.

Saving Processed Data

- The cleaned and encoded dataset is saved as "outliers_marked.csv".

```
[ ] df_outliers_marked.to_csv("outliers_marked.csv", index=False)
```

```
[ ] scaler = StandardScaler()
```

```
[ ] df_numeric_scaled = pd.DataFrame(scaler.fit_transform(df_numeric), columns=df_numeric.columns)
print("Standardized Dataframe:\n", df_numeric_scaled)
```

```
Standardized Dataframe:
   CGPA  Internships  Projects  Workshops/Certifications  \
0  -0.309343  -0.073889  -1.182822  -0.014598
1   1.877818  -1.575689   1.121526   1.091319
2  -0.621794  -0.073889  -0.030648   1.091319
3  -0.309343  -0.073889  -1.182822   1.091319
4   0.940464  -0.073889  -0.030648   1.091319
...    ...    ...    ...    ...
9995 -0.309343  -0.073889  -1.182822   1.091319
9996 -0.465568  -1.575689  -1.182822  -1.120516
9997  1.096689  -0.073889   1.121526  -1.120516
9998  1.877818  -1.575689   1.121526   1.091319
9999  1.096689  -1.575689  -1.182822  -0.014598
```

	AptitudeTestScore	SoftSkillsRating	SSC_Marks	HSC_Marks
0	-1.770910	0.184742	-0.782306	0.504368
1	1.292970	-0.787072	0.847618	0.840726
2	0.312528	1.156555	0.943496	0.616487
3	0.680194	0.184742	1.135251	0.616487
4	0.802749	0.427695	0.464106	1.513441
...
9995	-0.913024	-1.030025	1.518763	-0.953181
9996	1.292970	1.156555	1.422885	-0.841062
9997	-1.158134	1.156555	0.943496	0.728606
9998	0.925304	1.156555	0.176473	1.177083
9999	-1.648355	-1.272978	-0.686428	-0.953181

```
[10000 rows x 8 columns]
```


1. Normalization Technique:

- **MinMaxScaler()** is applied to scale all numerical features between 0 and 1.
- This transformation ensures that all values lie within a uniform range, preventing dominance by larger values.

2. Affected Features:

The normalized dataset includes:

- **CGPA**
- **Internships**
- **Projects**
- **Workshops/Certifications**
- **Aptitude Test Score**
- **Soft Skills Rating**
- **SSC Marks**
- **HSC Marks**

3. Purpose:

- Ensures that all numerical features contribute equally to the analysis/model.
- Helps in improving the performance of machine learning algorithms, particularly those that rely on distance-based calculations (e.g., KNN, SVM, Neural Networks).

What It Indicates

- The dataset is being prepared for machine learning by applying feature scaling.
- The placement prediction model will likely use this normalized data to enhance accuracy and avoid bias due to large variations in feature magnitudes.

```
normalizer = MinMaxScaler()  
df_numeric_normalized = pd.DataFrame(normalizer.fit_transform(df_numeric_scaled), columns=df_numeric.columns)  
print("Standardized and Normalized Dataframe:\n", df_numeric_normalized)
```

```
Standardized and Normalized Dataframe:  
   CGPA  Internships  Projects  Workshops/Certifications  \  
0   0.384615      0.5  0.333333      0.333333  
1   0.923077      0.0  1.000000      0.666667  
2   0.307692      0.5  0.666667      0.666667  
3   0.384615      0.5  0.333333      0.666667  
4   0.692308      0.5  0.666667      0.666667  
...   ...      ...      ...      ...  
9995  0.384615      0.5  0.333333      0.666667  
9996  0.346154      0.0  0.333333      0.000000  
9997  0.730769      0.5  1.000000      0.000000  
9998  0.923077      0.0  1.000000      0.666667  
9999  0.730769      0.0  0.333333      0.333333  
  
   AptitudeTestScore  SoftSkillsRating  SSC_Marks  HSC_Marks  
0           0.166667           0.777778   0.171429   0.709677  
1           1.000000           0.555556   0.657143   0.806452  
2           0.733333           1.000000   0.685714   0.741935  
3           0.833333           0.777778   0.742857   0.741935  
4           0.866667           0.833333   0.542857   1.000000  
...           ...      ...      ...      ...  
9995        0.400000           0.500000   0.857143   0.290323  
9996        1.000000           1.000000   0.828571   0.322581  
9997        0.333333           1.000000   0.685714   0.774194  
9998        0.900000           1.000000   0.457143   0.903226  
9999        0.200000           0.444444   0.200000   0.290323
```

[10000 rows x 8 columns]

Conclusion: In conclusion, this experiment highlighted the significance of proper data cleaning and preparation to ensure high-quality data. We addressed key issues such as missing values, irrelevant data, and outliers that could negatively impact the dataset and ultimately distort model performance. Missing values were managed through imputation or removal methods, irrelevant features were discarded, and outliers were carefully handled to avoid skewing the results. These steps were crucial for creating a dataset that is accurate and reliable for analysis.

Additionally, the dataset was scaled for uniformity, which is important to ensure that all features are treated equally. By applying standardization or normalization techniques, we ensured that the model could work efficiently without being influenced by differences in scale between variables. Overall, these data preparation techniques are fundamental in setting up a clean, well-structured dataset, which is essential for generating reliable model outcomes and drawing meaningful conclusions.

