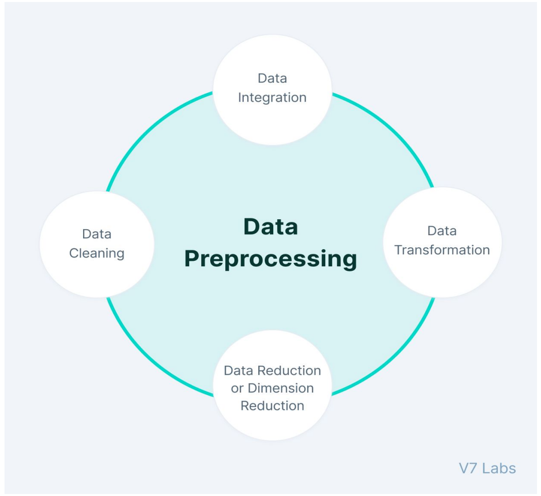
HSTU ML Enthusiasts

Class - 5
Data Preprocessing and Exploratory Data Analysis (EDA)

What is Preprocessing in Machine Learning?

Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models.



Some Preprocessing Techniques

- Data Cleaning.
- Dimensionality Reduction.
- Feature Engineering.
- Sampling Data.
- Data Transformation.
- Imbalanced Data.
- Feature Engineering
- Text Preprocessing
- Encoding Categorical Data

Importing Library & Reading Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
from sklearn import preprocessing
```

```
data = pd.read_csv('/content/Loan_Data.csv')
data.head()
```

Handle Missing Values

- Replacing With Arbitrary Value

```
data.fillna(method='ffill',limit=1)
```

```
data.fillna(method="bfill",limit=1)
```

- Filling missing values with 0

```
data.fillna(0)
```

- Delete Rows with Missing Values

```
data = data.dropna()
```

Handle Imbalance Dataset

- **Resampling** is a common technique used to address class imbalance in machine learning.
- **Ensemble techniques** involve combining multiple models to improve performance. This can be done by using techniques such as bagging, boosting, and stacking.
- **Evaluation metrics** for imbalanced data: Evaluation metrics such as precision, recall, and F1 score can be used to evaluate the performance of models on imbalanced data.
- **Data augmentation** involves creating additional data points by modifying existing data. This can be done by applying various transformations such as rotations, translations, and flips to the existing data.

Text Data Processing

Converting the text data into Numerical to get best performance in ML Algorithms.

- **Level encoding**
- **One Hot encoding**

Label encoding is a technique used in machine learning and data analysis to convert categorical variables into numerical format.

Gender	Level Encoded Data
Male	1
Female	0

One Hot Encoding

One-hot encoding is used to convert categorical variables into a format that can be readily used by machine learning algorithms. The basic idea of one-hot encoding is to create new variables that take on values 0 and 1 to represent the original categorical values which preserves actual weight of the value.

Education	One hot Encode
Graduated	1
Not Graduated	0

Drop Unwanted & Create New Column

```
# delete the column 'Gender'
data_drop = data.drop(labels='Locations', axis=1)
print(data_drop)
```

Create New Column

- To Reduce Dimension
- To Create New Target
- To Get Average of Few Columns

Exploratory Data Analysis (EDA)

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

Some Examples:

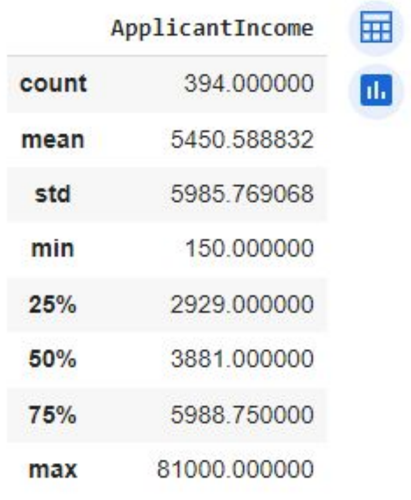
- Mean, Median, Mode, Max, Min, Count, Range, Sum, Standard Deviation
- Uni Variate Analysis
- Bi Variate Analysis
- Multivariate Analysis with Contour Plot
- Data Pattern & correlation between Variable
- Detect outlier & anomalies

Objective of EDA?

- List of outliers
- Estimates for parameters
- Uncertainties about those estimates
- List of all important factors
- Conclusions or assumptions as to whether certain individual factors are statistically essential
- Optimal settings
- A good predictive model

Mean, Median, Mode, Max, Min, Count, Range, Sum, Standard Deviation

```
data[data['Gender'] == 1][['ApplicantIncome']].describe()
```

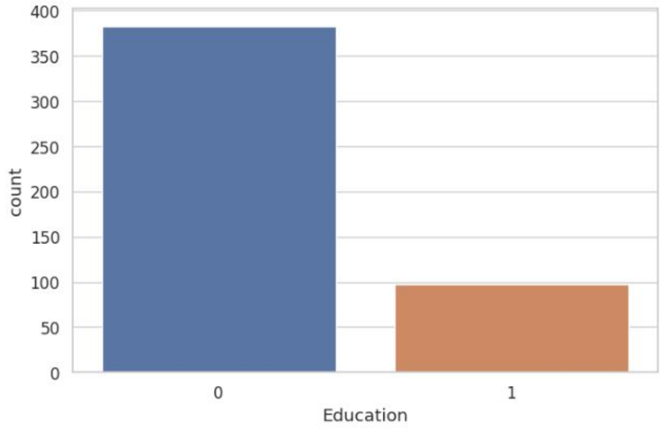


	ApplicantIncome
count	394.000000
mean	5450.588832
std	5985.769068
min	150.000000
25%	2929.000000
50%	3881.000000
75%	5988.750000
max	81000.000000

Univariate Analysis

Univariate Analysis is a type of data visualization where we visualize only a **single variable at a time**. Univariate Analysis helps us to analyze the distribution of the variable present in the data so that we can perform further analysis.

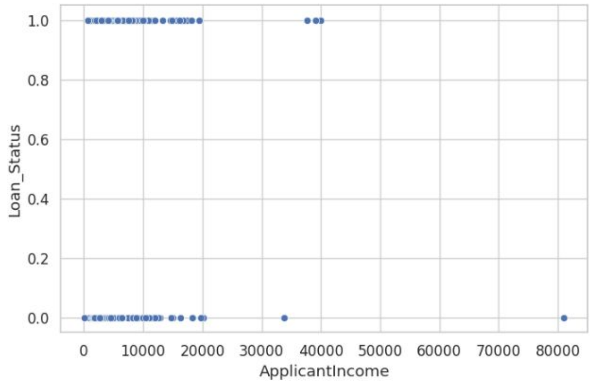
```
sns.countplot(data=data, x='Education')
plt.show()
```



Bivariate analysis

Bivariate analysis is the simultaneous analysis of two variables. It explores the concept of the relationship between two variable whether there exists an association and the strength of this association or whether there are differences between two variables and the significance of these differences.

```
sns.scatterplot(x=data['ApplicantIncome'], y=data['Loan_Status'])
```

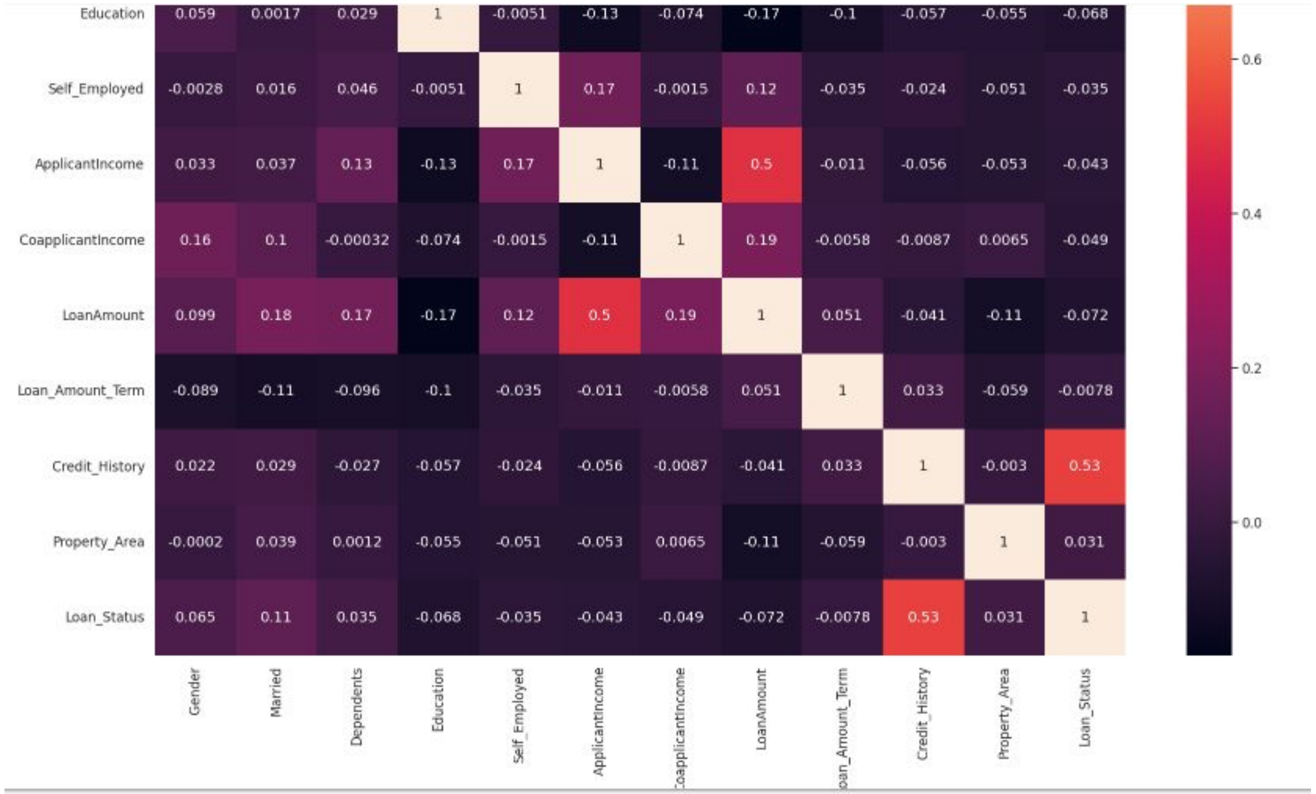


Multivariate Analysis

It is an extension of bivariate analysis which means it involves multiple variables at the same time to find correlation between them. Multivariate Analysis is a set of statistical model that examine patterns in multidimensional data by considering at once, several data variable.

```
plt.figure(figsize=(20,15))
sns.heatmap(data.corr(), annot=True)
```

Multivariate Analysis



Data Pattern & Correlation between Variable

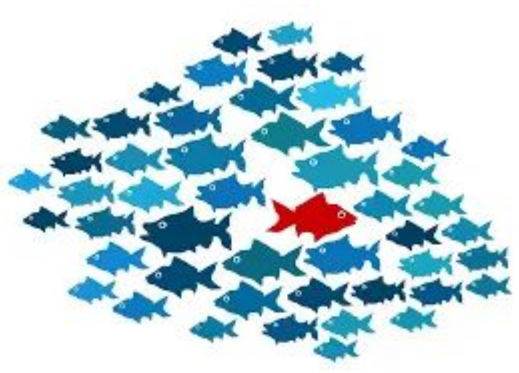
It can be useful in data analysis and modeling to better understand the relationships between variables. The statistical relationship between two variables is referred to as their correlation.

- Positive Correlation: both variables change in the same direction.
- Neutral Correlation: No relationship in the change of the variables.
- Negative Correlation: variables change in opposite directions.

Detect outlier with Machine Learning

In Data Science, an Outlier is an observation point that is distant from other observations. An Outlier may be due to variability in the measurement or it may indicate experimental error. Some Techniques:

- Mahalanobis Distance
- Cook’s Distance



Anomaly Detection with Machine Learning

Anomaly detection finds extensive use in a wide variety of applications such as fraud detection for credit cards, insurance or health care, intrusion detection for cyber-security, fault detection in safety critical systems, and military surveillance for enemy activities.

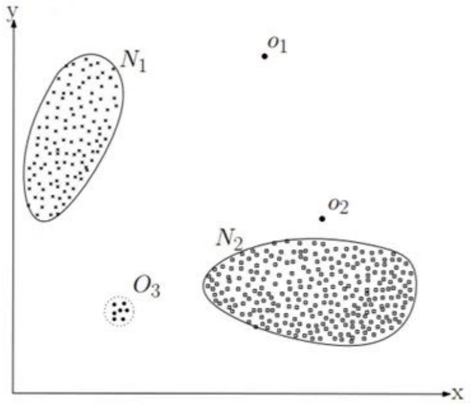


Fig. 1. A simple example of anomalies in a 2-dimensional data set.

Train Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(data.drop(['Loan_Status'],axis=1), data['Loan_Status'],
test_size=0.3)
```

Fit a ML model

```python
import xgboost as xgb

# Create a list of non-numeric columns to exclude
non_numeric_columns = ['Loan_ID']

# Exclude the non-numeric columns from X_train and X_test
X_train = X_train.drop(columns=non_numeric_columns)
X_test = X_test.drop(columns=non_numeric_columns)

# Create and fit the XGBoost model
model = xgb.XGBClassifier()  # You can choose the appropriate XGBoost model based on your task
model.fit(X_train, y_train)

# Evaluate the model
accuracy = model.score(X_test, y_test)
print("Accuracy:", accuracy)
```

Any Question?

Thank You