3. Machine Learning

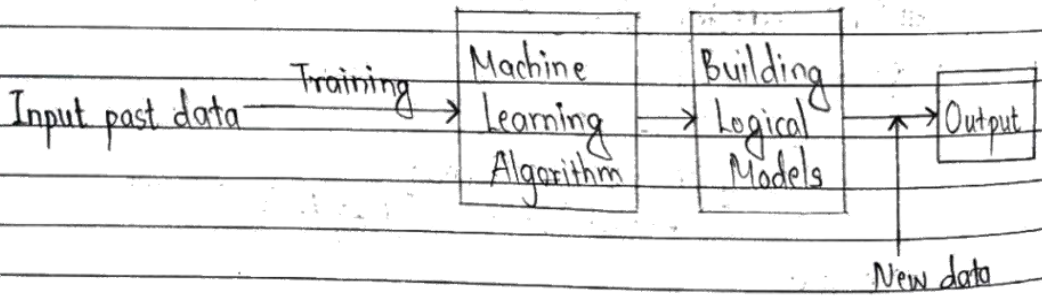
Date
Page 56

1. Introduction

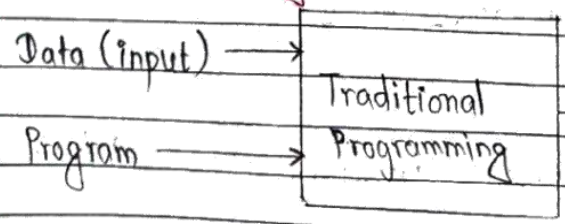
Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real world interactions.

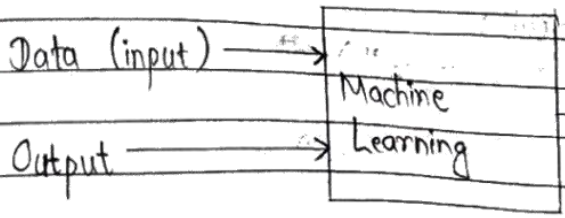
How Does Machine Learning Work ?

A machine learning system learns from historical data, builds the prediction models and whenever it receives new data, predicts the output for it.



Machine Learning V/s. Traditional Programming





Date
Page 57

Classification of Machine Learning

1. Supervised Learning → future predictions / recommendation sys
2. Unsupervised Learning → tasks related to groupism / grouping
3. Reinforcement Learning → used in game automation

Advantages of Machine Learning

- Easily identifies trends and patterns.
- No human intervention needed (automation)
- Continues improvement.
- Handling multi-dimensional and multi-variety data
- Wide applications

Disadvantages of Machine Learning

- Data acquisition
- Time and resources
- Interpretation of results
- High error-susceptibility

Uses of Machine Learning

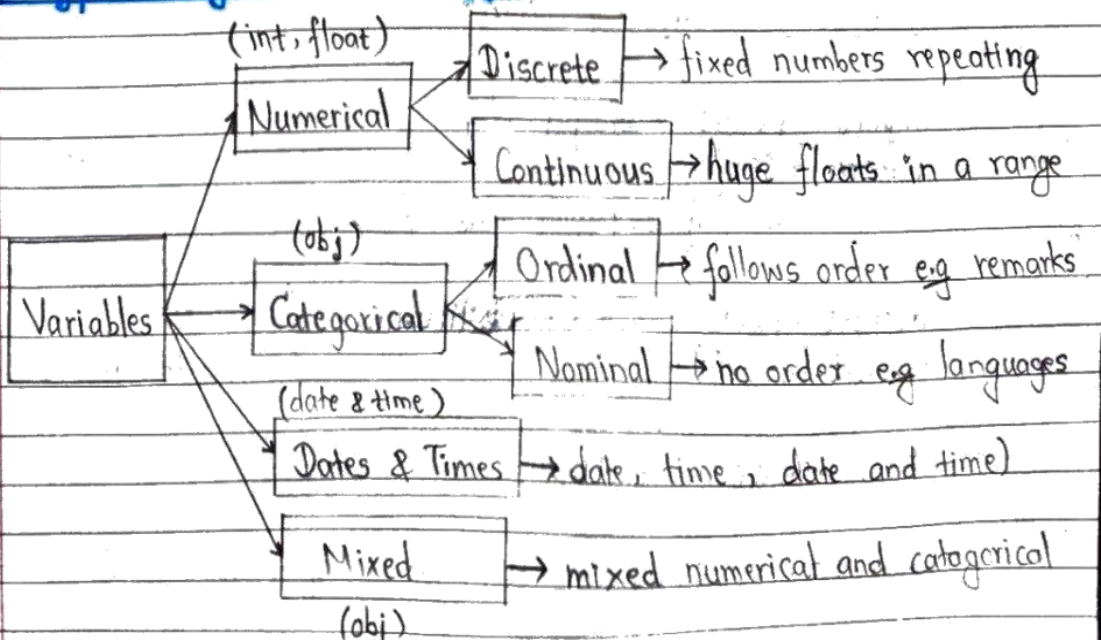
- Automatic language translation
- Medical diagnosis
- Stock market trading
- Online fraud detection
- Virtual personal assistant
- Email span and malware filtering
- Self driving cars
- Product recommendations
- Traffic prediction
- Speech recognition
- Image recognition

2. Road Map

- Programming language : Python (recommended), R
  - pandas → for data analysis
  - numpy → solve numerical problems
  - matplotlib → data analysis in graphical form
  - seaborn → data analysis in graphical form.
  - scipy → provides constants
  - scikit learn → machine learning models
  - tensorflow → deep learning
  - nltk → text processing
  - open cv → image processing
- Exploratory data analysis
- Feature engineering
  - Exploratory data analysis
  - Handling missing value
  - Handling outliers
  - Categorical encoding
  - Normalizing and standardization
- Feature selection
  - Correlation
  - Forward elimination
  - Backward elimination
  - Univariate selection
  - Random forest importance
  - Feature selection with decision trees

- Machine learning algorithms → regression and classification, clustring
- Linear, Logistic Regression, Decision Tree, Random Forest, K means
- Grid Search, Randomised Search, Hyperopot, Genetics algorithms.
- Dockers and Kubernetes
- Model deployments
- End to End ML Projects

3. Types of Variables



4. Data Cleaning

Data cleaning is the process of preparing data for analysis / ML / DL by removing and modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.

Steps of Data Cleaning

1. Handling missing data
2. Outlier detection and handling
3. Data scaling and transformation
4. Encoding categorical variables
5. Handling duplicates
6. Dealing with inconsistent data

1. Handling Missing Data

Finding out missing values:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset = pd.read_csv(r".\Students.csv")      #r means raw string
dataset.head(8)                               #. means current path

dataset.shape  # Displays (rows, columns)

dataset.shape[0] # Displays no. of rows, 1 for columns
```

```
dataset.isnull()  # Represent True if null value otherwise
# false in dataset

dataset.isnull().sum() # Provides total number of null
# values in each column.

dataset.isnull().sum().sum() # Provides total number of
# null values in dataset

# Similarly, notnull() can be used for non-null values
dataset.notnull().sum().sum()

(dataset.isnull().sum() / dataset.shape[0])* 100 # Shows
# total null values in each column in percentage.

(dataset.isnull().sum().sum()/(dataset.shape[0]* dataset.shape
[1]))* 100  # Shows total percentage of null value.

# for making a graphical representation
sns.heatmap(dataset.isnull())
plt.show()
```

Dropping missing values :

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset = pd.read_csv("Students.csv")
dataset.head(4)
```

```
dataset.shape  # Provides total number of rows and columns

dataset.isnull().sum()  # Provides total number of null
# values in each column.

# Lets plot the graphical visual
sns.heatmap(dataset.isnull())
plt.show()

# Droping a column with maximum null values.
dataset.drop(columns=["Gender"], inplace=True)

# Droping all the rows with null values.
dataset.dropna(inplace=True)

dataset.shape  # Provides remaining rows and columns

dataset.isnull().sum().sum()  # It must be zero.

# lets plot a graph.
sns.heatmap(dataset.isnull())
plt.show()
```

Imputing category data:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset = pd.read_csv("Students.csv")
```

Date Page 68

```
dataset.head(5)

dataset.isnull().sum()

dataset.fillna(5)  # fills all the missing values with 5
# Remember that we have many types of data with
# different data types so, filling same thing to all is
# not a convinent way.
# Category data can be filled using mode, backward
# filling or forward filling

dataset.info()  # Shows information related to dataset.

dataset.fillna(method="bfill")  # Backward filling

dataset.fillna(method="ffill")  # Forward filling.
# You can also choose axis, axis=0 is default.

# Filling mode in a particular column
dataset["Age"].fillna(dataset["Age"].mode()[0])

# Filling mode in all columns having categorical data.
for i in dataset.select_dtypes(include="object").columns:
    dataset[i].fillna(dataset[i].mode()[0])
# Use inplace=True if you want a permanent change.

# Lets plot a graph
sns.heatmap(dataset.isnull())
plt.show()
```

Imputing missing values using scikit-learn:

```
import pandas as pd

dataset = pd.read_csv("Students.csv")
dataset.head(5)

dataset.isnull().sum()

dataset.info()

dataset.select_dtypes(include="float64").columns
# Returns all the columns having float64 datatype.

columns = dataset.select_dtypes(include="float64").columns
# Store the columns in the variable as a list.

# Now imputing missing values using scikit-learn
from sklearn.impute import SimpleImputer

si = SimpleImputer(strategy="mean") # Create an object of class
si.fit_transform(dataset[columns]) # This will return the
# columns as in the form of array after filling null values

array = si.fit_transform(dataset[columns])
new_dataset = pd.DataFrame(array, columns=columns)
new_dataset # Transformed array to dataset.

new_dataset.isnull().sum() # No null values are present
# All the null values have been filled by means.
```

2. Outlier Detection and Handling

Outlier Detection :

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset = pd.read_csv ("Students.csv")
dataset.head ()

dataset.info () # Provides info about dataset

dataset.describe () # Describe dataset statistics

#Visual using boxplot
sns.boxplot (x="Age", data = dataset )
plt.show()

# We find no outliers in the data
# So lets create some outliers in the Age column by ourself
# for analysis by filling null values.
import random
dataset ["Age"].fillna(random.choice ([i for i in range (0,5)]
    + [i for i in range (80,95)]), inplace=True)

sns.boxplot (x = "Age", data = dataset)
plt.show()
# Now we can see outliers clearly in the boxplot
```

Outlier Handling:

Outlier Handling by Using Direct Method :

```
min_range = dataset["Age"].mean() - (3 * dataset["Age"].std())
        # min_range = mean - 3std
max_range = dataset["Age"].mean() + (3* dataset["Age"].std())
        #max_range = mean + 3std.
min_range , max_range

# Create a new dataset by removing outliers.
new_dataset = dataset[(dataset["Age"] <= max_range) &
                      (dataset["Age"] >= min_range)]
new_dataset.head(3)

sns.boxplot(x="Age", data = new_dataset)
plt.show()
```

" If some outliers are still remaining, avoid these because it can lead to huge data losses."

Outlier Removal Using IQR :

```
q1 = dataset["Age"].quantile(0.25) # First quantile
q2 = dataset["Age"].quantile(0.75) # Third quantile
q1, q3

iqr = q3 - q1  # Calculate IQR
iqr
```

```
#find range
min_range = q1 - (1.5 * iqr)
max_range = q3 + (1.5 * iqr)
min_range , max_range

#Remove the outliers
new_dataset = dataset [(dataset ["Age"] < max_range) &
                       (dataset ["Age"] > min_rage)]

sns.boxplot (x = "Age" , data = new_dataset)
plt.show()
```

"If some outliers are still remaining, avoid them because removing can lead to huge data losses"

Outlier Removal by Using Z Score :

- The Z-score method has same functionality as direct method.
- Mathematically,

$$Z\text{-score} = \frac{x - \mu}{\sigma}$$

where, μ = mean
σ = standard deviation

- Zrange = $\mu - 3\sigma$ to $\mu + 3\sigma$ i.e. -3 to $+3$.
- We need to select data having z-score within Zrange.

```
# Calculate z-score
z-score = (dataset["Age"] - dataset["Age"].mean()) /
          dataset["Age"].std()
z-score

dataset["Z Score"] = z-score #Adding a column to dataset

new-dataset = dataset[(dataset["Z Score"] < 3) &
              (dataset["Z Score"] > -3)] #Removes outliner.
new-dataset.head(4)

sns.boxplot(x="Age", data = new-dataset)
plt.show()
```

3. Data Scaling and Transformation

Feature Scaling (Standardization)

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset = pd.read_csv("Students.csv")
dataset.head(3)

dataset.isnull().sum()

dataset["Age"].fillna(dataset["Age"].mean(), inplace=True)
# fill null values
```

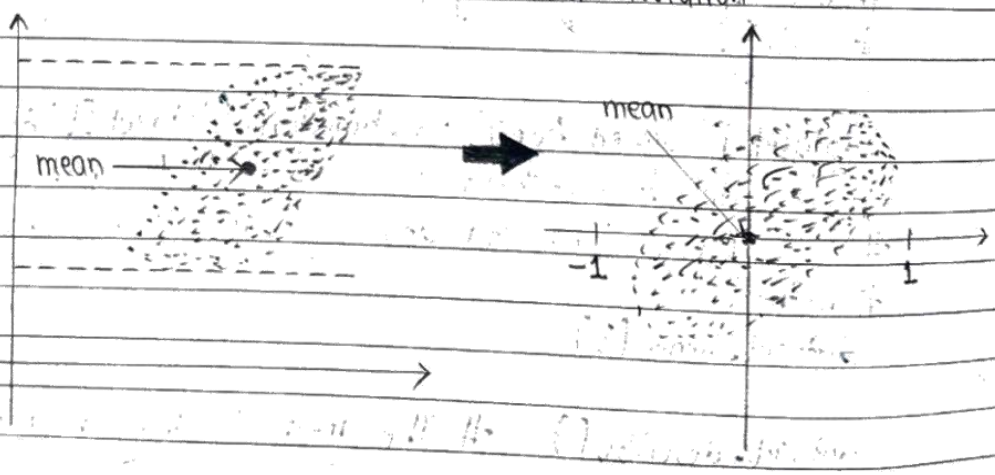
```
# Visualization.
sns.distplot(dataset["Age"])
plt.show()

dataset.describe  # Describe statistics of dataset

# Feature scaling using sklearn
from sklearn.preprocessing import StandardScalar

ss = StandardScalar()
ss.fit(dataset[["Age"]])  # Works upon two dimension
# dataset
# The fit(data) is used to compute the mean and std
# dev for a given feature to be used further for
# scaling.

dataset["Scaled Age"] =ss.transform(dataset[["Age"]])
# The transform(data) method is used to perform scaling
# using mean and std dev calculated using the .fit()
# method.
dataset.head(5)

dataset.describe()  # The mean is very near to zero

# For visualization
plt.figure(figsize=(10,7))
# Previous data graph
plt.subplot(1,2,1)
plt.title("Before")
sns.distplot(dataset["Age"])
# Scaled data graph
```

```
plt.subplot(1, 2, 2)
plt.title("After")
sns.distplot(dataset["Scaled Age"])
#Display plot
plt.show()
```

- **Standardization:** It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1.

- Mathematically,

$$X_{new} = \frac{X_i - X_{mean}}{\text{Standard deviation}}$$



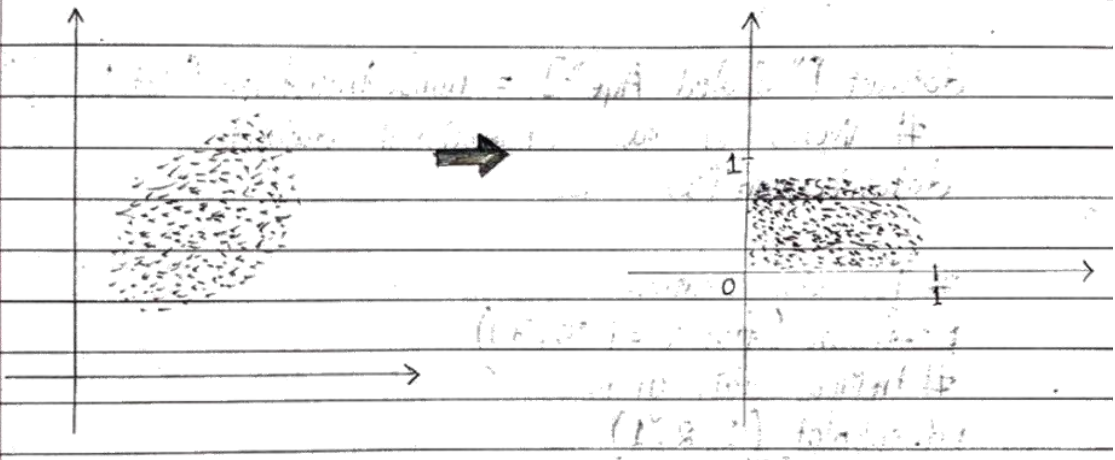
Feature Scaling (Normalization)

- **Normalization:** It is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1.

Date
Page 71

- It is also known as Min-Max scaling.
- Mathematically,

$$X_{new} = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$




```
dataset = pd.read_csv("Students.csv")
dataset.head(3)

dataset.isnull().sum()

dataset["Age"].fillna(dataset["Age"].mean(), inplace = True)
# Filling up null values

# See graphical nature
sns.distplot(dataset["Age"])
plt.show()

# Feature scaling (normalization)
from sklearn.preprocessing import MinMaxScalar
```

```
mms = MinMaxScalar()
mms.fit(dataset[["Age"]]) # Works upon two dimension dataset.
# The fit(data) method is used to compute the mean
# and std dev for a given feature to be used further
# for scaling.

dataset["Scaled Age"] = mms.transform(dataset[["Age"]])
  # Works on two dimensional dataset
dataset.head(3)

# For visualization
plt.figure(figsize=(10,7))
# Previous data graph
plt.subplot(1,2,1)
plt.title("Before")
sns.distplot(dataset["Age"])
# Scaled data graph
plt.subplot(1,2,2)
plt.title("After")
sns.distplot(dataset["Scaled Age"])
# Show graph
plt.show()
```

4. Encoding Categorical Variables

One Hot Encoding and Dummy Variables

- It is used when you have limited amount of categorical data.

```
import pandas as pd

dataset = pd.read_csv("Students.csv")
dataset.head()

dataset.isnull().sum() # Check for missing values

# Fill all the missing values with mode of the column
# which we want to encode.
dataset["Gender"].fillna(dataset["Gender"].mode()[0],
                         inplace = True)
dataset["Enrollment Status"].fillna(dataset["Enrollment
        Status"].mode()[0], inplace = True)
# Now the data is ready for encoding
```

Using Pandas : get_dummies()

```
en_data = dataset[["Gender", "Enrollment Status"]]
# Getting all the columns that needs to encode.

encoded_data = pd.get_dummies(en_data)
# Encoding our data (one hot encoding)
encoded_data

encoded_data.info() # After encoding we have our data
# in boolean form

# But we need data in numerical form
encoded_data.apply(lambda x : x.astype(int)) # Convert
# boolean data to numerical form.
```

```
# Now convert it in Data frame
numeric_data = encoded_data.apply(lambda x: x.astype(int))
pd.DataFrame(numeric_data, columns = numeric_data.columns)
```

Using scikit-learn : OneHotEncoder

```
from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder()
ohe.fit_transform(en_data) # Provides an output in form
of sparse matrix
#Sparse matrix is a matrix in which elements are filled
#with 0 and 1.
# The fit_transform() method does both fits and
#transform.

# To get the data in form of array
ohe.fit_transform(en_data).toarray()

# But we need it in the form of data frame
ar = ohe.fit_transform(en_data).toarray()
pd.DataFrame(ar, columns = encoded_data.columns)

# If your data is too large and want to drop the
first column you can proceed as
ohe = OneHotEncoder(drop = "first")
ar = ohe.fit_transform(en_data).toarray()
pd.DataFrame(ar, columns = [encoded_data.columns[1],
                            encoded_data.columns[3]])
# Because first columns of both categorical data was
# removed i.e. columns[0] and columns[2] are removed
```

Label Encoding

↳ It is performed on nominal data (data having no order sequence i.e. has no connections)

```
import pandas as pd
df = pd.DataFrame({"Name" : ["Mukesh", "Nikesh",
                    "Ashok", "Sandip", "Subha"]})
df

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit_transform(df["Name"]) # The fit_transform()
# method does both fits and transform.

# If we need to add encoded data column to our
# Data Frame
df["Encoded Name"] = le.fit_transform(df["Name"])
df
```

Take some real world example and apply concept of label encoding

```
dataset = pd.read_csv("Students.csv")
dataset.head(3)

label_enc = LabelEncoder()
label_enc = fit(dataset["Faculty"])
# The fit(data) method is used to compute the
```

```
# mean and std dev for a given feature to be
# used further for scaling.

label_enc.transform(dataset["Faculty"])
# The transform(data) method is used to perform
# scaling using mean and std dev calculated by using
# the .fit() method.

# If you want to replace the data with encoded data
dataset["Faculty"] = label_enc.transform(dataset["Faculty"])
dataset
```

Ordinal Encoding

- It is performed on ordinal data (data having order sequence i.e. has connections)

With Scikit-learn :

```
import pandas as pd

df = pd.DataFrame({"Size": ["s", "l", "xl", "m", "l",
                            "s", "xl", "m", "s", "xl", "l"]})
df.head(5)

data_order = [["s", "m", "l", "xl"]] # Two dimensional
# list for order of data.

from sklearn.preprocessing import OrdinalEncoder
```

```
oe = OrdinalEncoder(categories = data_order)
oe.fit(df[["Size"]])

oe.transform(df[["Size"]])

# If you want to add encoded data to Data Frame
df["Encoded With Scikit-learn"] = oe.transform(df
                                  [["Size"]])
df
```

With Map Function:

```
data_order = {"s":0, "m":1, "l":2, "xl":3}
# You can decide the numbers accordingly

df["Size"].map(data_order) # Ordinal encoding using
# map()

# If you want to save the encoded data to Data Frame
df["Encoded With Map function"] = df["Size"].map(
                                  data_order)
df
```

Performing ordinal encoding in large dataset:-

```
dataset = pd.read_csv("Students.csv")
dataset.head(4)

dataset["Level"].isnull().sum() # Checking for null
# values in Level column
```

```
dataset["Level"].fillna(dataset["Level"].mode()[0],
                        inplace=True) # Filling up null values

dataset["Level"].unique() # Returns all the unique
# values present in the column

encode_data_order = [["Graduate", "Post Graduate", "Phd"]]
# order encoding

from sklearn.preprocessing import OrdinalOrder

ordinal_encoder = OrdinalEncoder(categories=encode_data_order)
dataset["Level"] = ordinal_encoder.fit_transform(dataset[[
                                "Level"]])
dataset.head(5)
```

5. Handling Duplicates

- Content is called duplicate if two rows are exactly same.
- Having same data in some columns doesn't mean it is duplicate.

```
import pandas as pd

data = {
    "Name": ["Mukesh", "Nikesh", "Binod", "Ashok", "Subha",
             "Nikesh", "Subham"],
    "Physics": [56, 67, 87, 45, 33, 67, 34],
```

```
    "Chemistry" : [56, 64, 34, 56, 45, 64, 89],
    "Biology" : [64, 67, 76, 45, 56, 67, 32],
}
df = pd.DataFrame(data)
df
```

```
df.duplicated() # Returns the duplicated rows
```

```
df["Duplicated"] = df.duplicated() # Adding duplicated
# data in data frame
df
```

```
# Remove the duplicated column which we have added
df.drop("Duplicated", axis = 1, inplace = True)
df
```

```
df.drop_duplicates(keep = "first", inplace = True)
df
```

6. Dealing With Inconsistent Data

Replace and Data Type Change

```
import pandas as pd

dataset = pd.read_csv("Students.csv")
dataset.head(3)
```

```
dataset.info() # Get info about dataset
```

```
dataset.isnull().sum() # Get the null values

# Fill null values in Phone Number column.
dataset["Phone Number"].fillna(dataset["Phone Number"]
        .mode()[0], inplace=True)

dataset["Phone Number"].value_counts() # Get count
# for each unique value present in dataset column

# Lets convert the data into the numerical data i.e.
# remove - sign
dataset["Phone Number"] = dataset["Phone Number"].str
        .replace("-","")
dataset.head(5)

# Convert data into numerical datatype
dataset["Phone Number"] = dataset["Phone Number"]
                .astype("int64")
dataset.head(5)

dataset.info() # You will see the change in datatype
```

5. Function Transformation

↳ Transformation of data by using mathematical functions:
- → If you have non-normal distribution data
- → If your data pattern is very high
- → Want to change the units of data

Pros : It can be used multiple times on a same data to meet our need like getting normally distributed data and wide range of using functions as we can use any type of function.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt.

dataset = pd.read_csv("Students.csv")
dataset.head(4)

dataset.isnull().sum()

# Lets drop all the rows with null values in the age
# column
dataset["Age"].dropna(axis=0, inplace=True)

sns.distplot(dataset["Age"])
plt.show()

# The graph is not normally distributed so lets try to
# make it a distributed graph
# First lets check by removing outliers
q1 = dataset["Age"].quantile(0.25)
q3 = dataset["Age"].quantile(0.75)
iqr = q3 - q1
min_range = q1 - (1.5 * iqr)
max_range = q3 + (1.5 * iqr)
min_range, max_range
```

```
dataset = dataset[(dataset["Age"] >= min-range) &
                  (dataset["Age"] <= max-range)]
dataset.head(3)

sns.distplot(dataset["Age"])
plt.show()
"""No much progress found"""

# So lets try by us using function transformer
from sklearn.preprocessing import FunctionTransformer
import numpy as np

ft = FunctionTransformer(func=np.log1p) # Natural logarithm
# of 1+x , element wise

ft.fit(dataset["Age"])

dataset["Transformed Age"] = ft.transform(dataset["Age"])
dataset.head(4)

# for visualization
plt.figure(figsize=(10,7))
# Previous data graph
plt.subplot(1,2,1)
plt.title("Before")
sns.distplot(dataset["Age"])
# Scaled data graph
plt.subplot(1,2,2)
plt.title("After")
sns.distplot(dataset["Transformed Age"])
plt.show()
```

```
# You can also use your own function like this
fun_transformer = FunctionTransformer(fun = lambda x:
                                      x** 2)
dataset["My Transformed Age"] = fun_transformer.fit_
                                transform(dataset["Age"])
dataset.head(3)

plt.subplot(1,2,1)
plt.title("Before")
sns.distplot(dataset["Age"])
# Scaled data graph
plt.subplot(1,2,2)
plt.title("After")
sns.distplot(dataset["My Transformed Age"])
plt.show()
```

6. Feature Selection Technique

- Select the feature which are important
- Drop the features which are not important

1. Forward Elimination

i. Begin with an empty model (no features)

ii. Add features iteratively. In each iteration, add one feature that improves the model the most.

iii. Evaluate the model's performance using a choosen metric (accuracy, AUC, etc.)

iv. Add the feature that results in the best improvement to the model. Repeat the process adding one feature at a time.

v. Continue adding features until adding more features doesn't improve the model's performance.

```
import pandas as pd
from mlxtend.feature_selection import SequentialFeatureSelector

data = {
    "Gulcose" : [148, 85, 183, 89, 137, 116, 78, 115, 197,
                 125, 110, 168, 139, 189, 166, 100],
    "Pressure" : [72, 44, 66, 64, 40, 74, 50, 0, 70, 96, 72,
                  74, 80, 60, 72, 0],
    "Skin Thickness" : [35, 29, 0, 23, 35, 0, 32, 0, 45,
                        0, 0, 0, 0, 23, 19, 0],
    "BMI" : [33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31,
             35.3, 30.5, 0, 37.6, 38, 27.1, 30.1,
             25.8, 30],
    "Age" : [50, 31, 32, 21, 33, 30, 26, 29, 53, 54,
             30, 34, 57, 59, 51, 36],
    "Outcome": [1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1,
                1]
}
dataset = pd.DataFrame(data)
dataset.head(3)
```

```
# Seperate x-axis and y-axis data.
x = dataset.iloc[:, :-1]
y = dataset["Outcome"]

# Now we will use classical analysis
from sklearn.linear_model import LogisticRegression

# Initialize the estimator
lr = LogisticRegression()

# Initialize the SequentialFeatureSelector.
sfs = SequentialFeatureSelector(estimator=lr, k_features=5,
                                forward=True)

# fit the selector
sfs.fit(x, y)

# Print the given features name
sfs.feature_names

# Print the selected features
sfs.k_feature_names_

# The top-k accuracy score
sfs.k_score_
```

2. Backward Elimination

1. Start with all features: Begin with a model that includes all possible features.

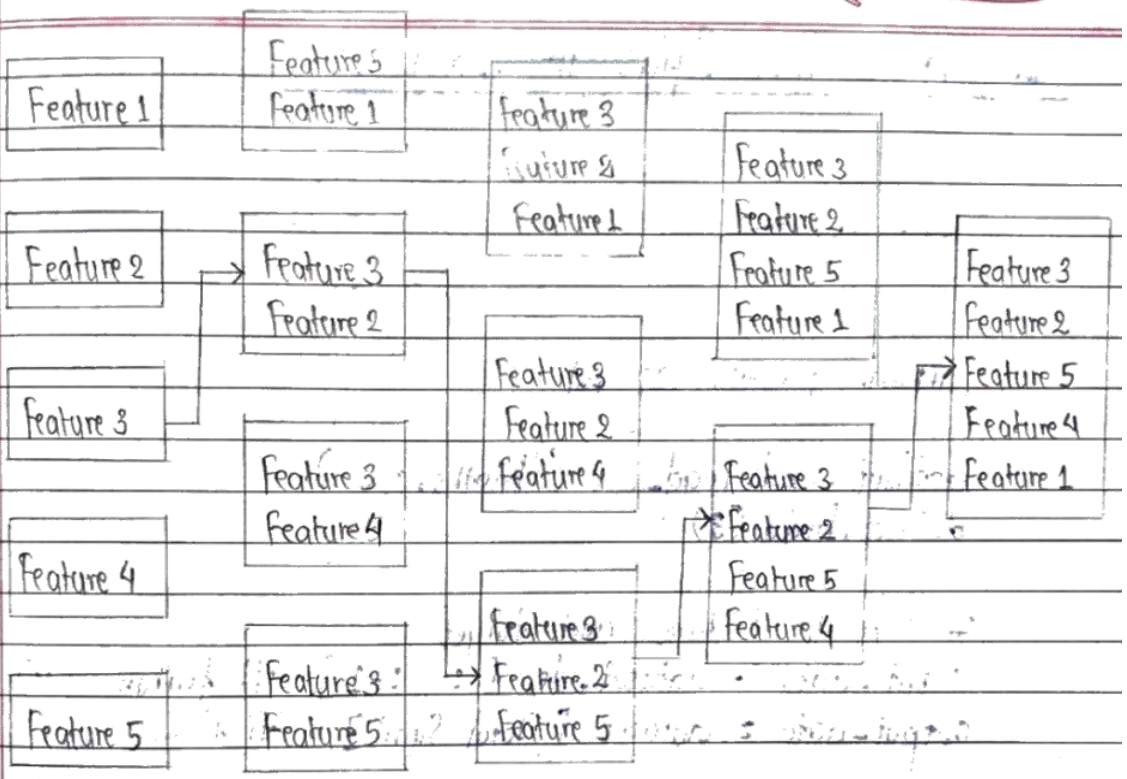
ii. Fit the model : Fit the model using all the features.

iii. Evaluate feature significance : Evaluate the significance of using a choosen metric (e.g. p value in regression)

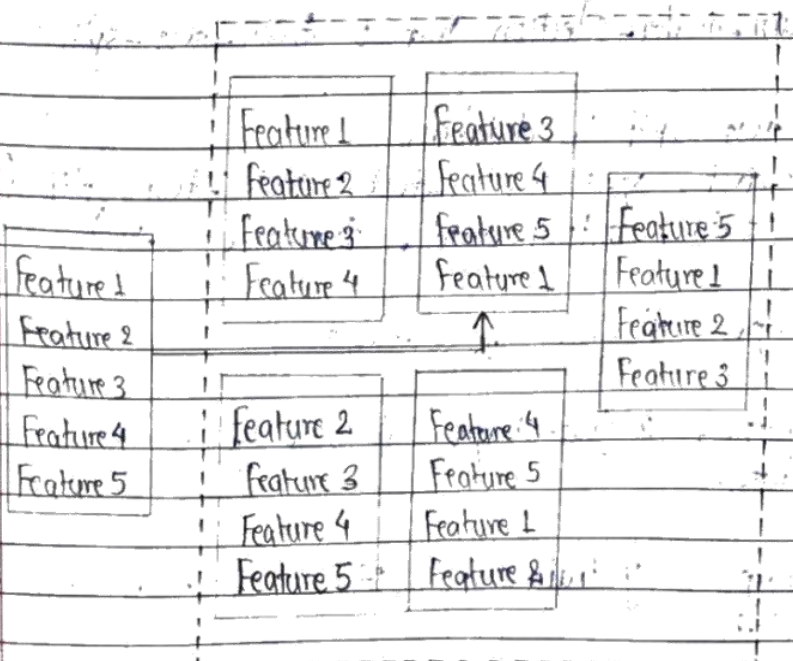
iv. Remove the least significant feature : Identify the feature with the least significance (highest p-value) and remove it from the model.

v. Repeat : Refit the model without the removed feature and repeat the process until all the features are significant..

```
# Initialize the Sequential Feature Selector
fsb = SequentialFeatureSelector(estimator=lr, k_features=5,
                                forward=False)

# fit the selector
fsb.fit(x,y)

# Prints the given feature names
fsb.feature_names

# Prints the selected feature names
fsb.k_feature_names_

# The top-k accuracy score
fsb.k_score_
```



FORWARD ELIMINATION



It stops here because there is no improvement found by our model on moving onword. It also happens in forward elimination.

BACKWARD ELIMINATION