



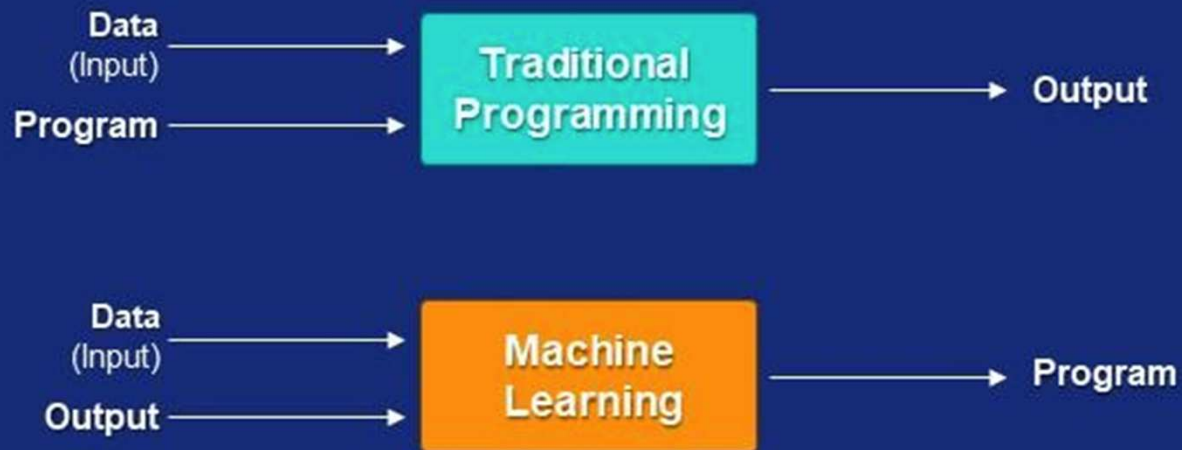
Evolution of Machine Learning and its application

Submitted By:
Amit Dua
Phd, Politechnika Slaska

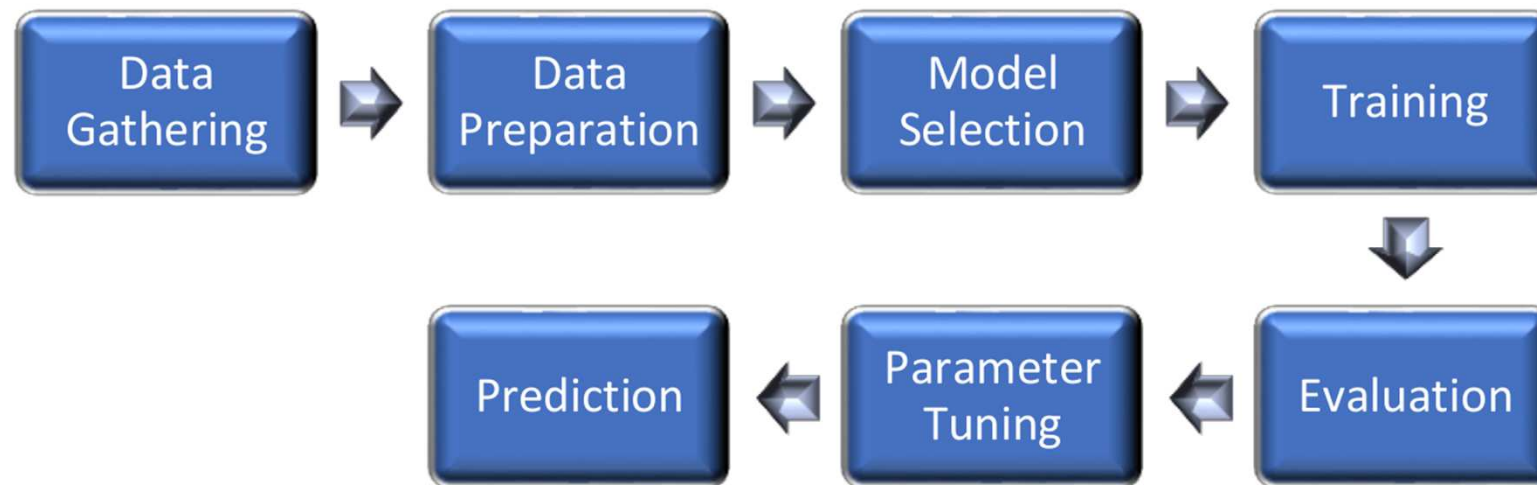
Summary

- .Machine Learning is all about finding patterns in data. Idea is to replace “human writing code” with a “human supplying data”.
- .Train our machines to use the data known as sample data to make predictions or valuable decisions without being explicitly programmed to do so.

What is Machine learning

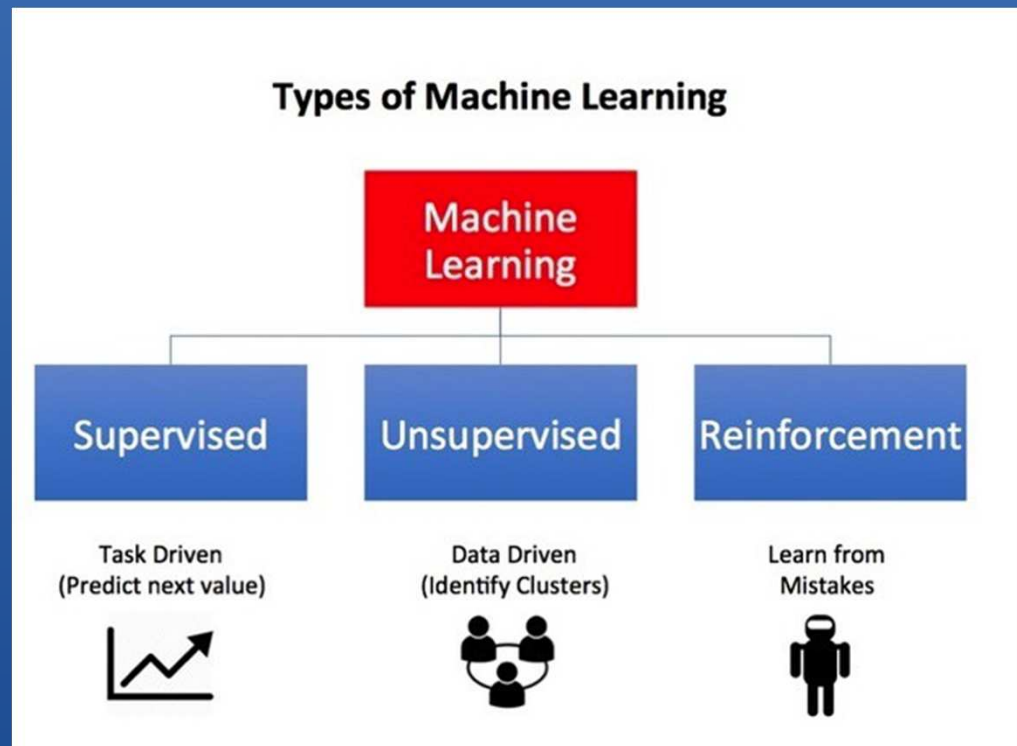


Working Model of Machine Learning



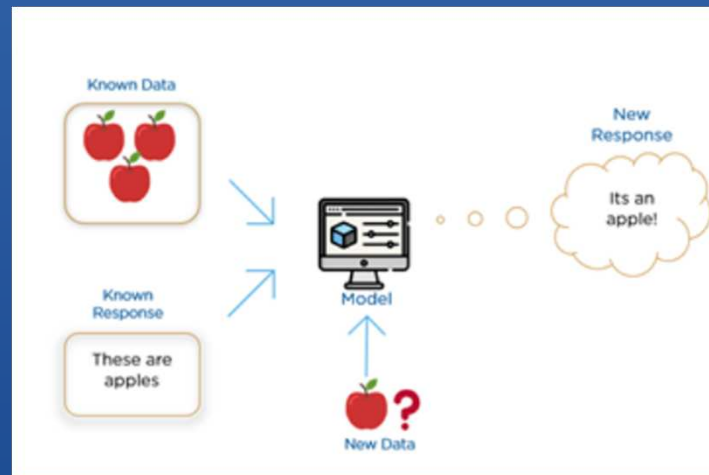
Types of Machine Learning

- .Supervised
- .Unsupervised
- .Reinforcement

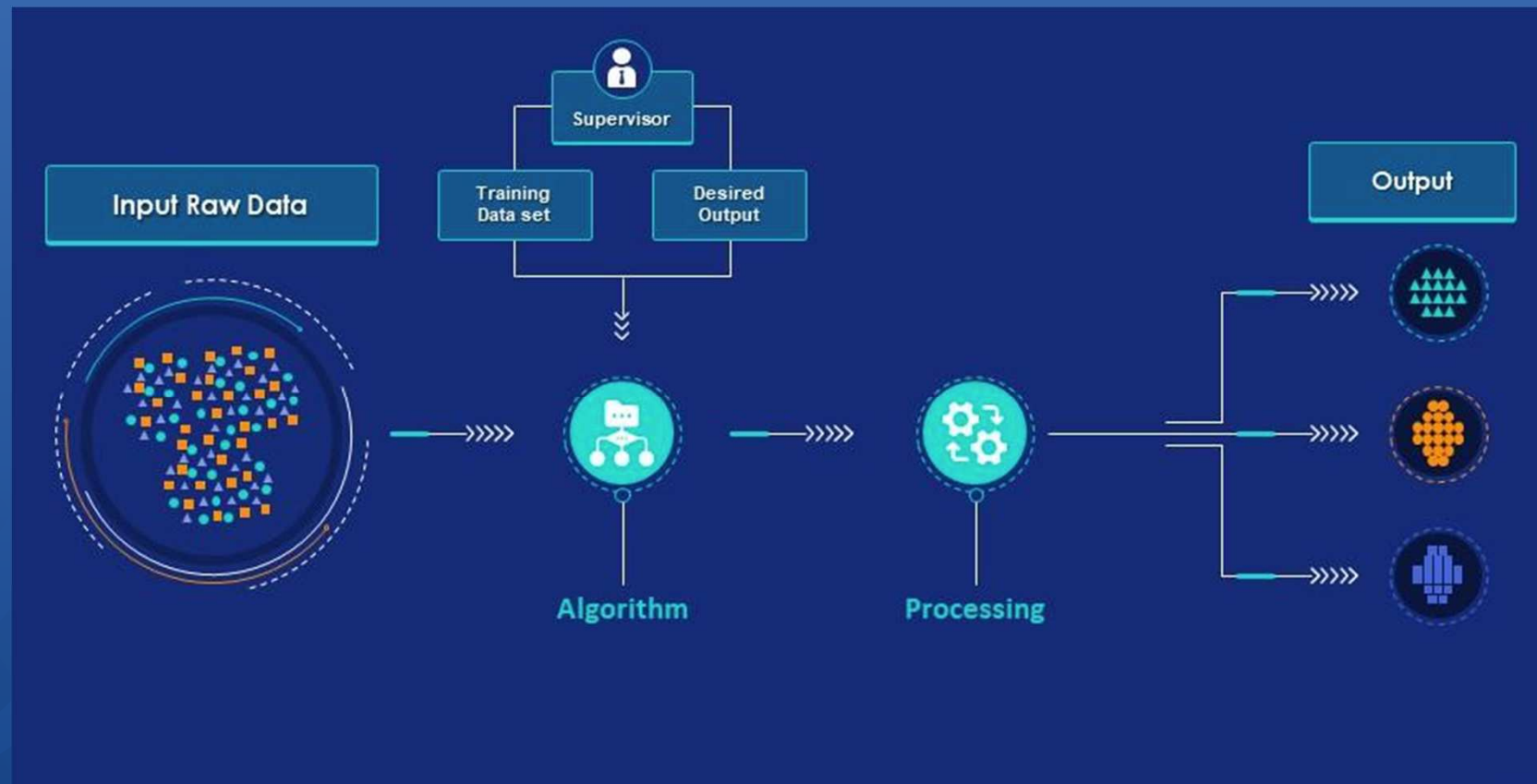


Supervised Learning

- Provide the machine learning algorithm with the “labeled data”
- Algorithm learns based on provided labeled data.



Supervised learning workflow



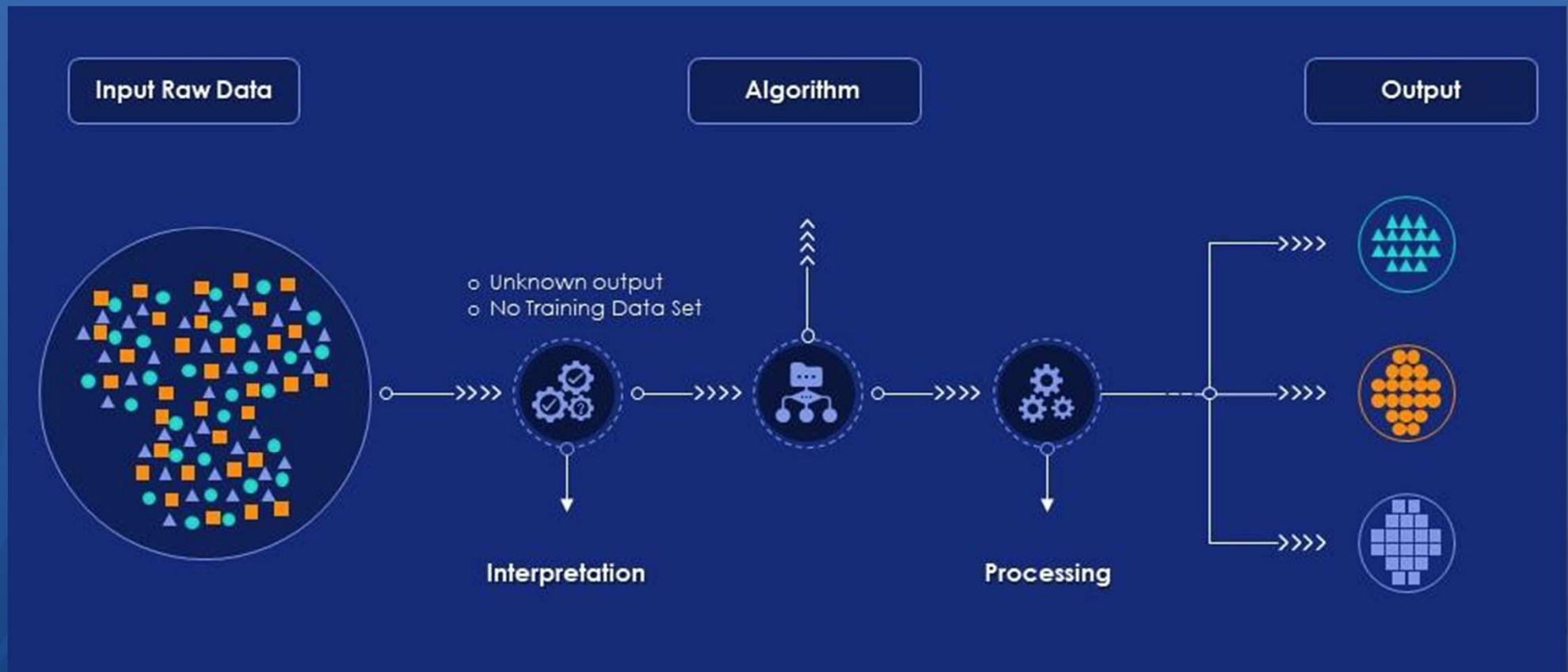
Common examples of Supervised Learning

- .Email Spam Detection (classification of Data)
- .Fraud Detection
- .Image Classification (Face recognition in Facebook)
- .Score predication (Regression approach)

Unsupervised Learning

- .No Labeled Data
- .Widely Used as most of the data is unlabeled.
- .Algorithm is fed with unlabeled data and the algorithm groups, clusters or organizes the data (to find the pattern)
- .Observe and learn from the pattern identified by the machine.
- .Unsupervised learning is Data driven (outcome is controlled by the data and its formatting)

Unsupervised Learning workflow



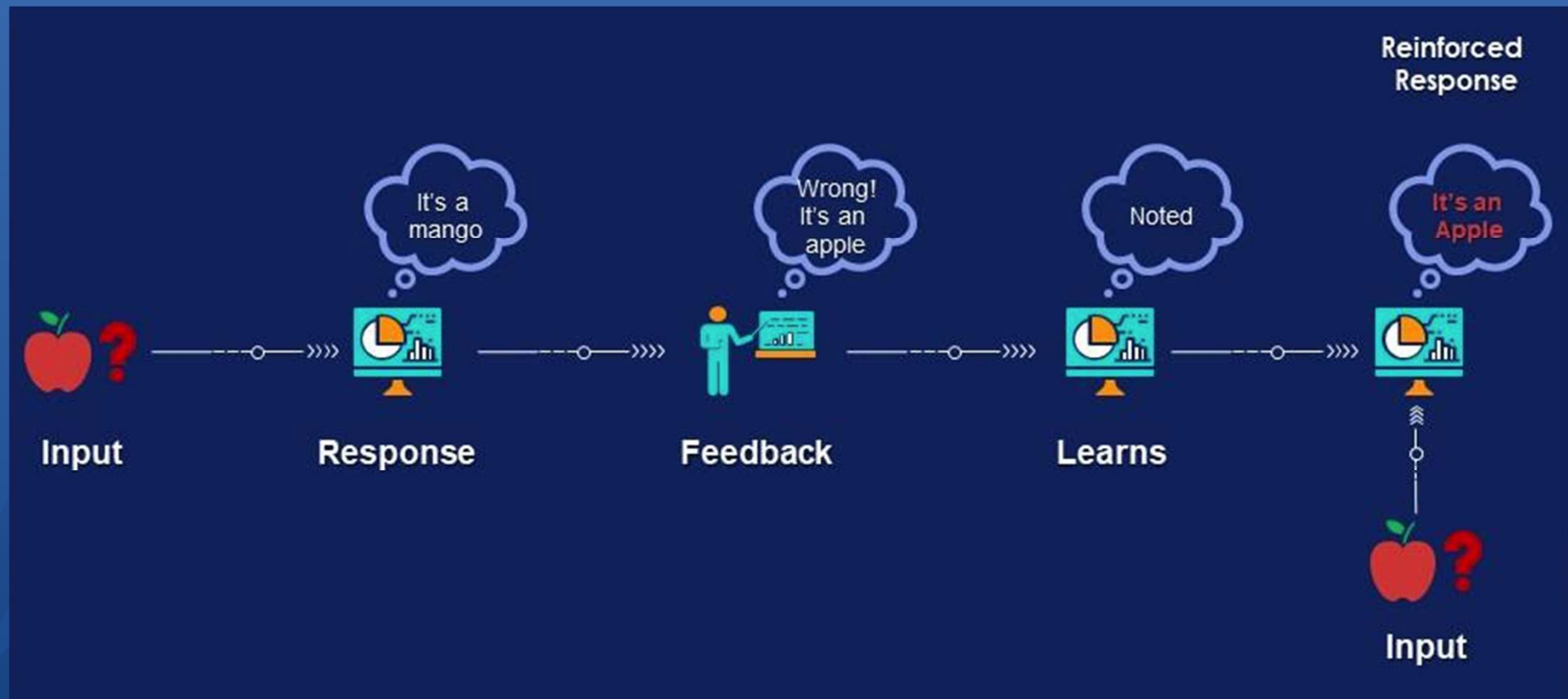
Applications of Unsupervised Learning

- .Media Recommendation systems- Netflix/YouTube
- .Recommendation of buying together based on learning of buying habits
- .Clustering of data such as user logs/issues, research papers of similar domain

Reinforcement Learning

- .The algorithm learns from the mistakes.
- .Our feedback to the algorithm response with positive and negative signals reinforces the algorithm to make good predictions after learning from mistakes.

What is Reinforcement learning



Application of Reinforcement Learning

- .Gaming
- .Finance sector
- .Inventory management

Machine Learning example

•**Title:** Multi-linear regression method to predict the chances of a student to get an admission into university

•**Aim:** For student to get an admission in a reputed university is a dream, since reputed universities look for students who have good grades as well as other important factor such as student GRE score, TOFEL or IELTS Score which is an English language test for student who want to get admission in reputed universities, university ranking, State of purpose(SOP), Letter of recommendation.

•We will predict chances of getting an admission in to a university based on the students grades and other important documents can help student to choose university according to their grades. For this purpose, we used a multi-linear regression method to predict the chances of getting an admission into an university

What is Regression Analysis

- process of predicting a Label (or Dependent Variable) based on the features (Independent Variables)
- time series modeling and finding the causal effect relationship between the variables and forecasting.
- fit a curve/line to the data points, in such a manner that the differences between the distance of the actual data points from the plotted curve/line is minimum

Program

#Let's start with importing necessary libraries

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge,Lasso,RidgeCV, LassoCV, ElasticNet, ElasticNetCV, LinearRegression
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
sns.set()
```

#load the input data

```
data =pd.read_csv('/content/drive/MyDrive/Admission_Prediction.csv')
data.head()
```

Showing Input Data top 5

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337.0	118.0	4.0	4.5	4.5	9.65	1	0.92
1	2	324.0	107.0	4.0	4.0	4.5	8.87	1	0.76
2	3	NaN	104.0	3.0	3.0	3.5	8.00	1	0.72
3	4	322.0	110.0	3.0	3.5	2.5	8.67	1	0.80
4	5	314.0	103.0	2.0	2.0	3.0	8.21	0	0.65

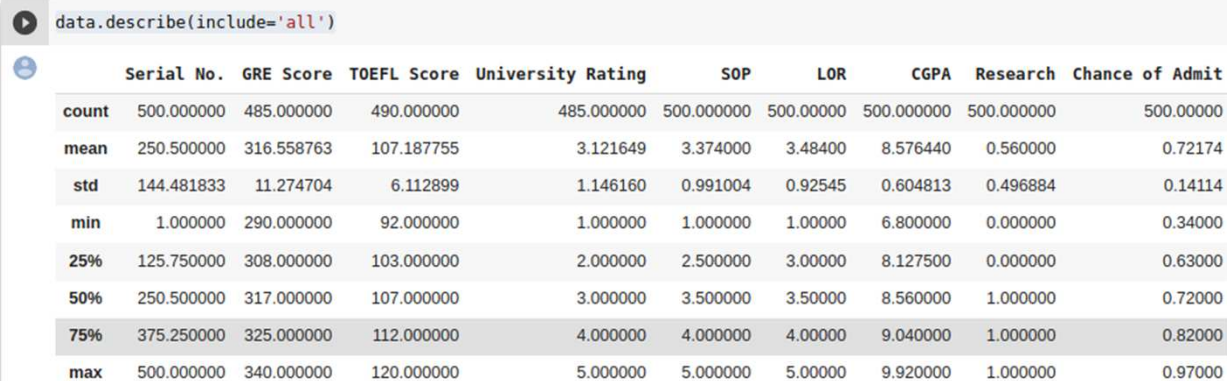
Continued

Let's create a function to create adjusted R-Squared

```
def adj_r2(x,y):  
    r2 = regression.score(x,y)  
    n = x.shape[0]  
    p = x.shape[1]  
    adjusted_r2 = 1-(1-r2)*(n-1)/(n-p-1)  
    return adjusted_r2
```

#Exploratory data analysis(EDA) of dataset

```
data.describe(include='all')
```



	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	485.000000	490.000000	485.000000	500.000000	500.000000	500.000000	500.000000	500.000000
mean	250.500000	316.558763	107.187755	3.121649	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.274704	6.112899	1.146160	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

Continued.

Finding missing values and filling

```
[ ] #Checking for missing value in the dataset
data.isna().value_counts()
```

Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	
False	False	False	False	False	False	False	False	False	460
	True	False	False	False	False	False	False	False	15
	False	False	True	False	False	False	False	False	15
		True	False	False	False	False	False	False	10

dtype: int64

Since we can observe there are few missing values exists in the dataset, we will first insert some missing value using fillna() function

```
[ ] data['University Rating'] = data['University Rating'].fillna(data['University Rating'].mode()[0])
data['TOEFL Score'] = data['TOEFL Score'].fillna(data['TOEFL Score'].mean())
data['GRE Score'] = data['GRE Score'].fillna(data['GRE Score'].mean())
```

Now the data looks good and there are no missing values. Also, the first column is just serial numbers, so we don't need that column. Let's drop it from data and make it more clean.

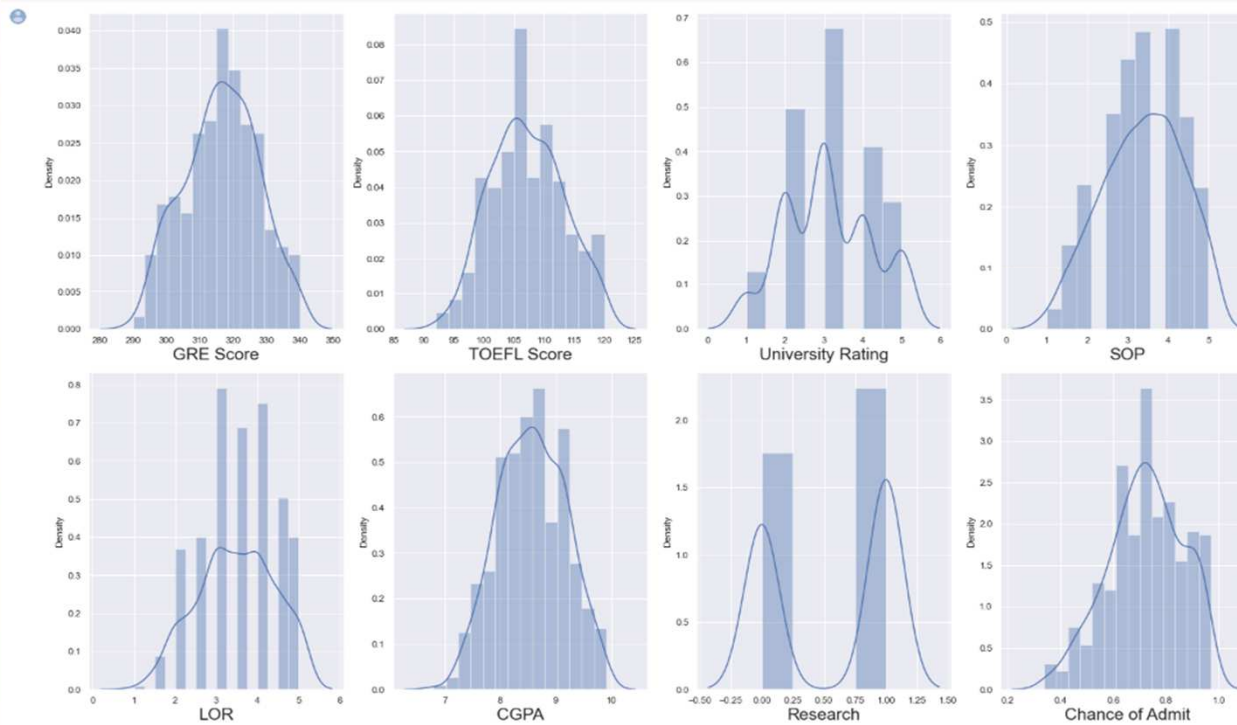
```
data = data.drop(columns = ['Serial No.'])
data.head()
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337.000000	118.0	4.0	4.5	4.5	9.65	1	0.92
1	324.000000	107.0	4.0	4.0	4.5	8.87	1	0.76
2	316.558763	104.0	3.0	3.0	3.5	8.00	1	0.72
3	322.000000	110.0	3.0	3.5	2.5	8.67	1	0.80
4	314.000000	103.0	2.0	2.0	3.0	8.21	0	0.65

Let us see the data distribution

```
# let's see how data is distributed for every column
plt.figure(figsize=(20,25), facecolor='white')
plotnumber = 1

for column in data:
    if plotnumber<=16 :
        ax = plt.subplot(4,4,plotnumber)
        sns.distplot(data[column])
        plt.xlabel(column,fontsize=20)
        #plt.ylabel('Salary',fontsize=20)
        plotnumber+=1
plt.tight_layout()
```



Defining Dependent and Independent Variables and visualizing relationship

Separating dependent and independent variable

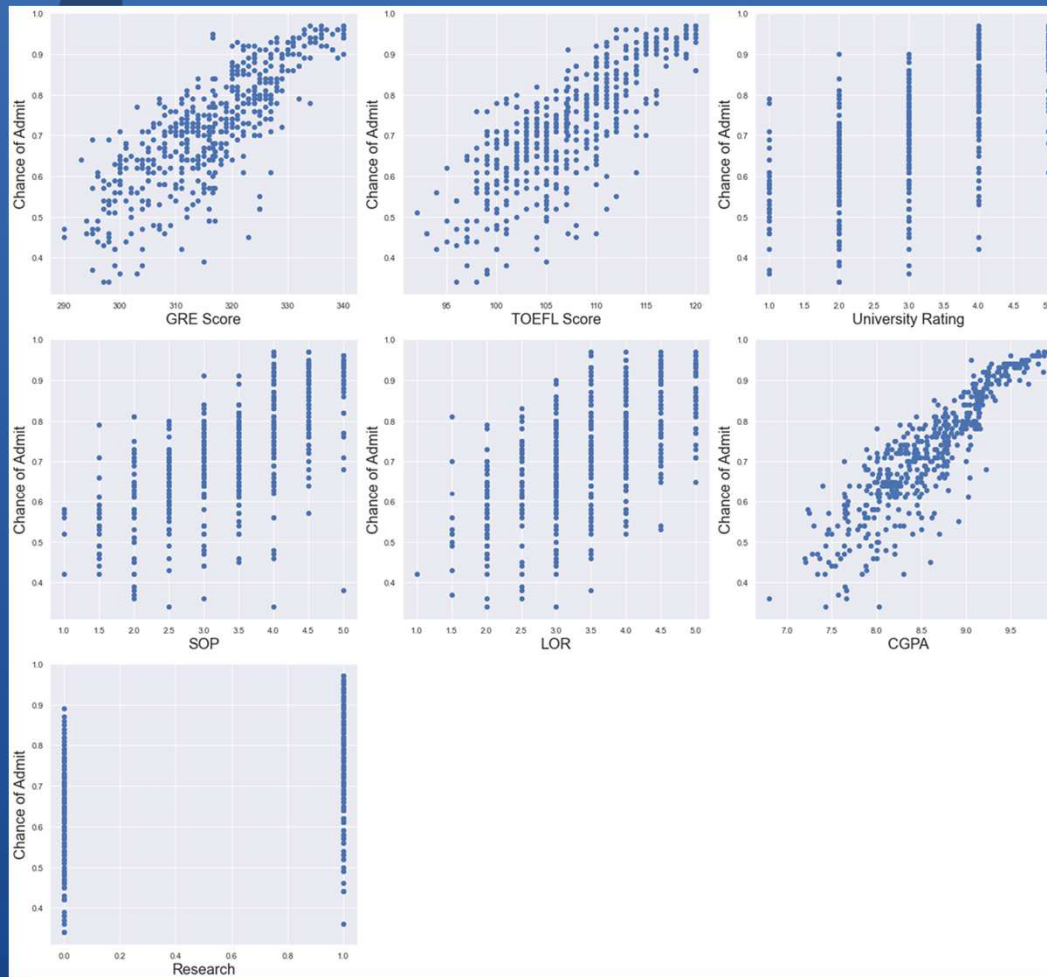
```
y = data['Chance of Admit']  
X = data.drop(columns = ['Chance of Admit'])
```

Let's visualize the data and analyze the relationship between independent and dependent variables:

```
plt.figure(figsize=(20,30), facecolor='white')  
plotnumber = 1
```

```
for column in X:  
    if plotnumber <= 15 :  
        ax = plt.subplot(5,3,plotnumber)  
        plt.scatter(X[column],y)  
        plt.xlabel(column,fontsize=20)  
        plt.ylabel('Chance of Admit',fontsize=20)  
        plotnumber += 1  
plt.tight_layout()
```

Visualizing relationship between dependent and independent variables



Great, the relationship between the dependent and independent variables is clear.

Let's move ahead and check for multi-collinearity.

Variance inflation factor

.The Variance Inflation Factor (VIF) is a measure of colinearity among predictor variables within a multiple regression. It's calculated by taking the the ratio of the variance of all a given model's betas divide by the variane of a single beta if it were fit alone

Standarizing data

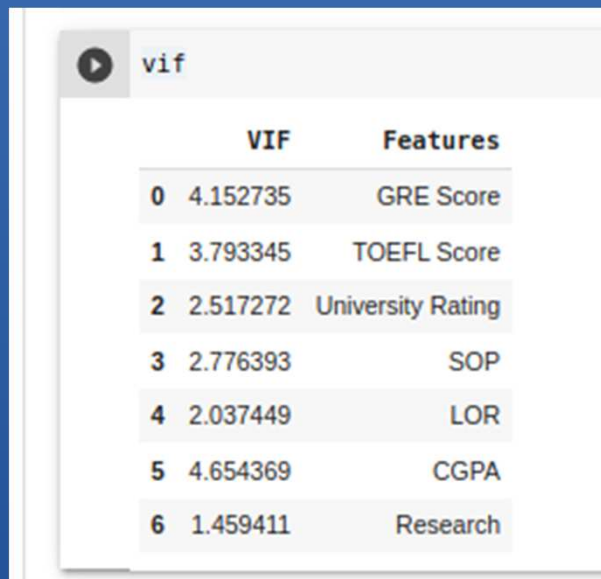
```
scaler =StandardScaler()  
X_scaled = scaler.fit_transform(X)
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor  
variables = X_scaled
```

```
# we create a new data frame which will include all the VIFs  
# note that each variable has its own variance inflation factor as this measure is variable specific (not model specific)  
# we do not include categorical values for mulitcollinearity as they do not provide much information as numerical ones do
```

```
vif = pd.DataFrame()  
# here we make use of the variance_inflation_factor, which will basically output the respective VIFs  
vif["VIF"] = [variance_inflation_factor(variables, i) for i in range(variables.shape[1])]  
# Finally, I like to include names so it is easier to explore the result  
vif["Features"] = X.columns
```


Checking VIF numbers



The screenshot shows a Jupyter Notebook cell with the label 'vif'. It contains a table with two columns: 'VIF' and 'Features'. The table lists VIF values for seven features: GRE Score, TOEFL Score, University Rating, SOP, LOR, CGPA, and Research. The VIF values are 4.152735, 3.793345, 2.517272, 2.776393, 2.037449, 4.654369, and 1.459411 respectively. All values are below 5, indicating no severe multicollinearity.

	VIF	Features
0	4.152735	GRE Score
1	3.793345	TOEFL Score
2	2.517272	University Rating
3	2.776393	SOP
4	2.037449	LOR
5	4.654369	CGPA
6	1.459411	Research

Here, we have the correlation values for all the features. As a thumb rule, a VIF value greater than 5 means a very severe multicollinearity. We don't have any VIF greater than 5.

Great. Let's go ahead and use linear regression and see how good it fits our data. But first, let's split our data in train and test.

Splitting Data into Test and Training set and calculate r2 score

Splitting the dataset into train and test set

```
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y,test_size = 0.25,random_state=355)
```

```
regression = LinearRegression()
```

```
regression.fit(x_train,y_train)
```

```
regression.score(x_train,y_train) = 0.8415250484247909
```

```
adj_r2(x_train,y_train) = 0.8385023654247188
```

#Our r2 score is 84.15% and adj r2 is 83.85% for our training set., so looks like we are not being penalized by use of any feature

#Let's check how well model fits the test data.

```
regression.score(x_test,y_test) =0.7534898831471066
```

```
adj_r2(x_test,y_test) = 0.7387414146174464
```

Results

We trained regression model to predict whether a student will get an admission into an university based on the different features, Since there were many feature we fit multi regression model. We accessed the trained model performance using R^2 statistic and adjusted R^2 statistic. After training regression model it achieved 84.15% R^2 score 83.85% adjusted R^2 score on training dataset. And on the test data-set, it achieved 75.34% R^2 score and 73.87% adjusted R^2 score, which is a significant score.

Applications Areas

Social Media and Internet

Image Video Classification

Speech Recognition

Language processing

Sentiment Analysis

Email Intelligence

Medicine and Healthcare

Cancer Detection

Illness Prediction

Drug discovery

Personalized care

Aged care

Automation

Image detection

Motion Tracking

Automated
Emergency
management

Signal Recognition

Product based Organizaton/E-commerce

Product
Recommendation

Dynamic Pricing

Sales Forecasting

Inventory
Management

Media & Entertainment

Recommendation
media

Content based
Search

Security

Video surveillance

Cyber Security

Event Prediction

Face Recognition

Self Driving Car

Object Detection

Lane Tracking

Traffic signal
detection