SESSION 9

1. Machine Learning - Introduction

- Machine Learning is a fascinating field that helps computers learn how to perform tasks without being explicitly programmed.
- Machine learning is a subset of artificial intelligence that involves training algorithms to make predictions or decisions based on data.
- The goal of machine learning is to develop algorithms that can learn and improve from experience without being explicitly programmed.
- It involves the use of statistical and mathematical techniques to create models that can recognize patterns in data and make predictions or decisions based on those patterns.

Machine Learning Example

Imagine you are teaching a robot how to play a game. In traditional programming, you would have to write out every single step the robot needs to take in order to play the game. But with machine learning, the robot can learn on its own by playing the game over and over again and getting better with each try.

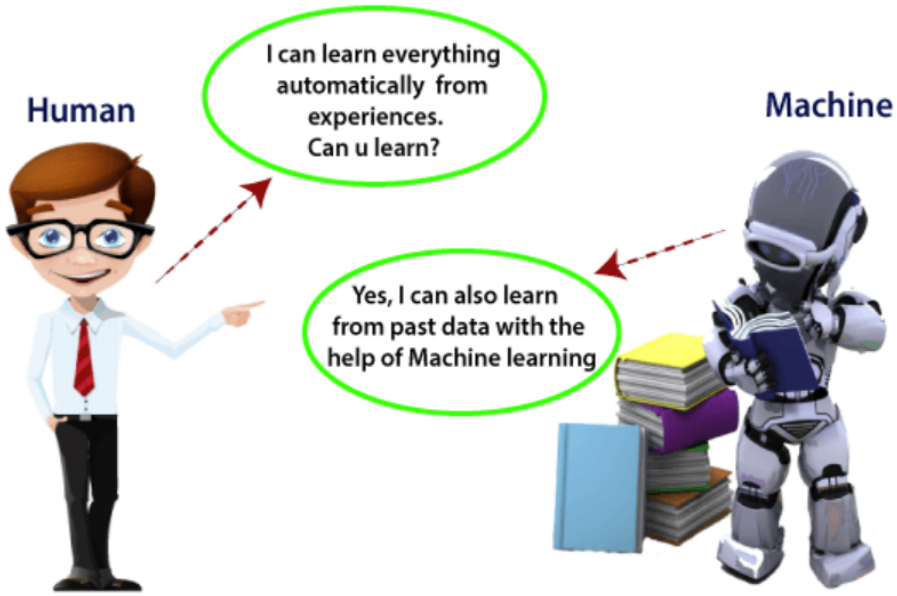


Image Credits: AISSMS Poly

2. Types of Machine Learning

There are four main types of machine learning: **Supervised learning, Unsupervised learning, Semi-supervised learning, Reinforcement learning**

Supervised learning

- In supervised learning, the computer is given a set of inputs (like pictures of animals) and a set of corresponding outputs (like what animal each picture shows).
- The computer then tries to learn the relationship between the inputs and outputs so that it can correctly identify animals in new pictures it has never seen before.
- The goal is for the algorithm to learn to predict the correct output for new inputs that it has not seen before.
  There are two main types of supervised learning:

1) Classification 2)Regression

UnSupervised learning

- The computer is given a set of inputs but no corresponding outputs. The computer then tries to find patterns or similarities within the inputs on its own.
- Unlike supervised learning, there are no known output values provided to the algorithm. The goal of unsupervised learning is to find patterns, structure, or relationships in the data on its own.
- The goal is for the algorithm to discover hidden relationships and structure in the data.

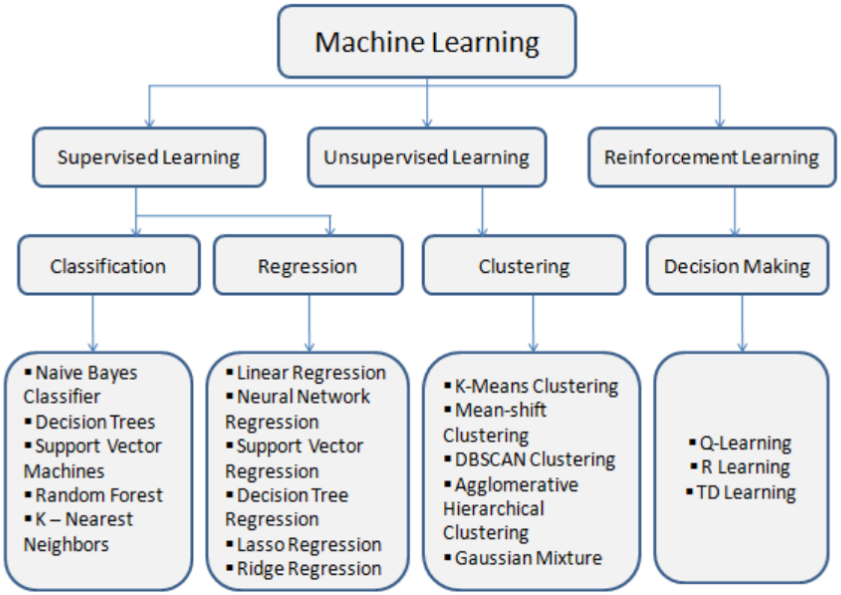


Image Credits: AISSMS Poly

3. Types of Supervised Learning:

Classification:

- The computer program is trained to sort things into different categories. `For example,` imagine you wanted to teach a computer program to tell the difference between pictures of dogs and pictures of cats. You would give the program lots of pictures of dogs and cats and label each one as either a dog or a cat. The program would then use this labeled data to learn how to recognize the difference between a dog and a cat.

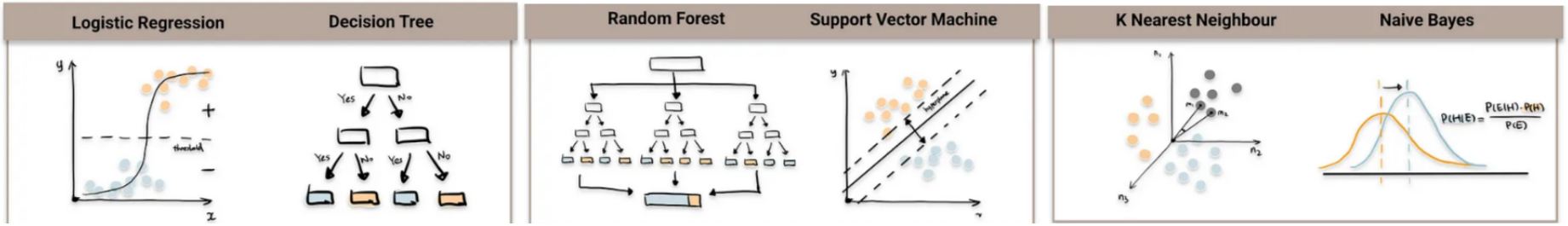


Image Credits: AISSMS Poly

Regression:

- In regression, a computer program is trained to make predictions based on patterns in data. `For example,` imagine you wanted to teach a computer program to predict the temperature outside based on the time of day. You would give the program lots of data about what the temperature was at different times of day and let it learn the patterns. The program would then use these patterns to make predictions about what the temperature might be at a specific time in the future.

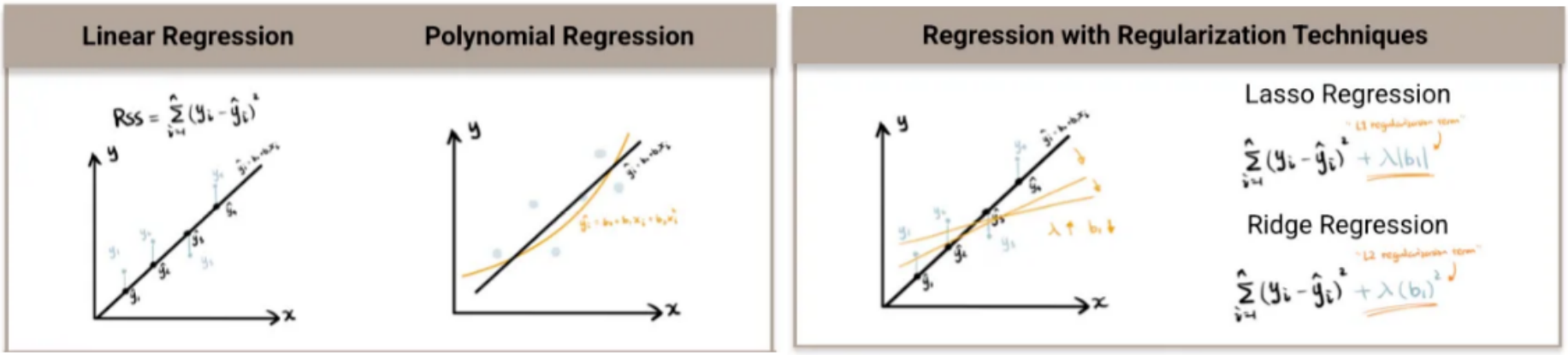


Image Credits: AISSMS Poly

4. Linear Regression - Introduction

Linear regression is a way to find a straight line that best fits a set of data points. For example, if you were given a bunch of data points that showed the relationship between a person's age and their height, you could use linear regression to find a straight line that best describes this relationship. Once you have this line, you can use it to make predictions about a person's height based on their age.

- One of the most straightforward and well-liked Machine Learning methods is linear regression. It is a statistical technique for performing predictive analysis. Predictions are made using linear regression for continuous/real/numeric variables like sales, salary, age, and product price, among others.
- Linear regression is a statistical method used to model the relationship between two variables, where one variable (called the dependent variable) is predicted based on the values of one or more independent variables.
- It assumes that there is a linear relationship between the independent variables and the dependent variable.
- The linear regression model represents this relationship as a straight line, which is defined by an equation in the form of `y = mx + b` . Here, y is the dependent variable, x is the independent variable, m is the slope of the line, and b is the intercept.
- The goal of linear regression is to find the values of m and b that best fit the data. This is typically done by minimizing the sum of the squared differences between the predicted values of y and the actual values of y, which is called the residual sum of squares (RSS).
- Linear regression can be used for both `Simple regression` , where there is only one independent variable and `Multiple regression` , where there are multiple independent variables.

Applications of Linear Regression:

1. Finance:

- The capital price asset model uses linear regression to analyze and quantify the systematic risks of an investment.

1. Trend lines:

- A trend line represents the variation in quantitative data with the passage of time (like GDP, oil prices, etc.).
- These trends usually follow a linear relationship. Hence, linear regression can be applied to predict future values.

1. Economics:

- Linear regression is the predominant empirical tool in economics.
- For example, it is used to predict consumer spending, fixed investment spending, inventory investment, purchases of a country's exports, spending on imports, the demand to hold liquid assets, labor demand, and labor supply.

5. Simple Linear Regression

In order to provide a basic understanding of linear regression, we start with the most basic version of linear regression, i.e. Simple linear regression.

- Simple linear regression is an approach for predicting a response using a single feature.
- It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

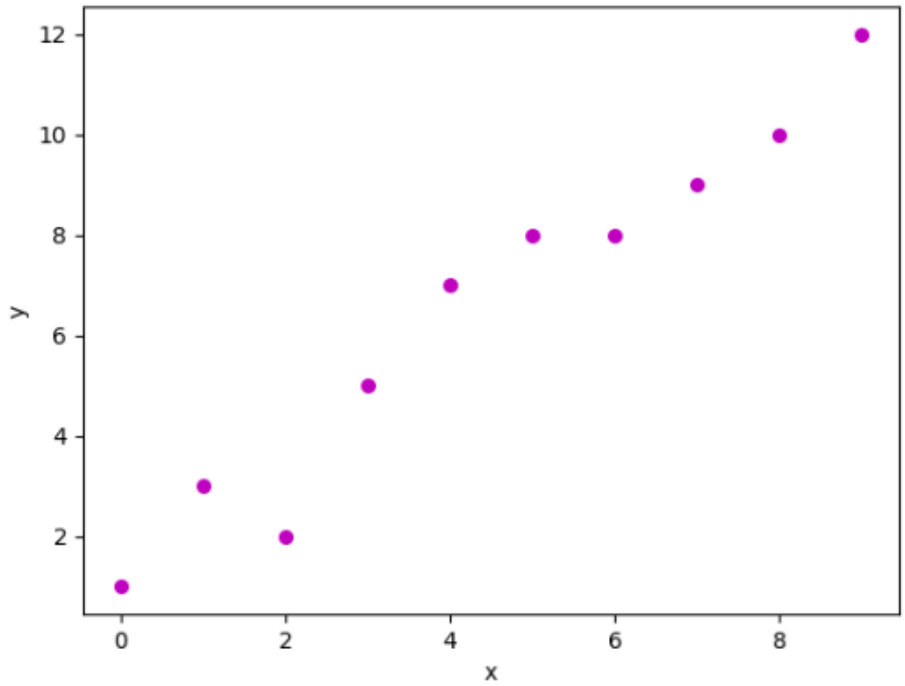
> Let us consider a dataset where we have a value of response y for every feature x

x	0	1	2	3	4	5	6	7	8	9
y	1	3	2	5	7	8	8	9	10	12

For generality, we define:

- x as feature vector, i.e `x = [x_1, x_2, …., x_n],`
- y as response vector, i.e `y = [y_1, y_2, …., y_n]`
- for n observations (in above example, `n=10`).

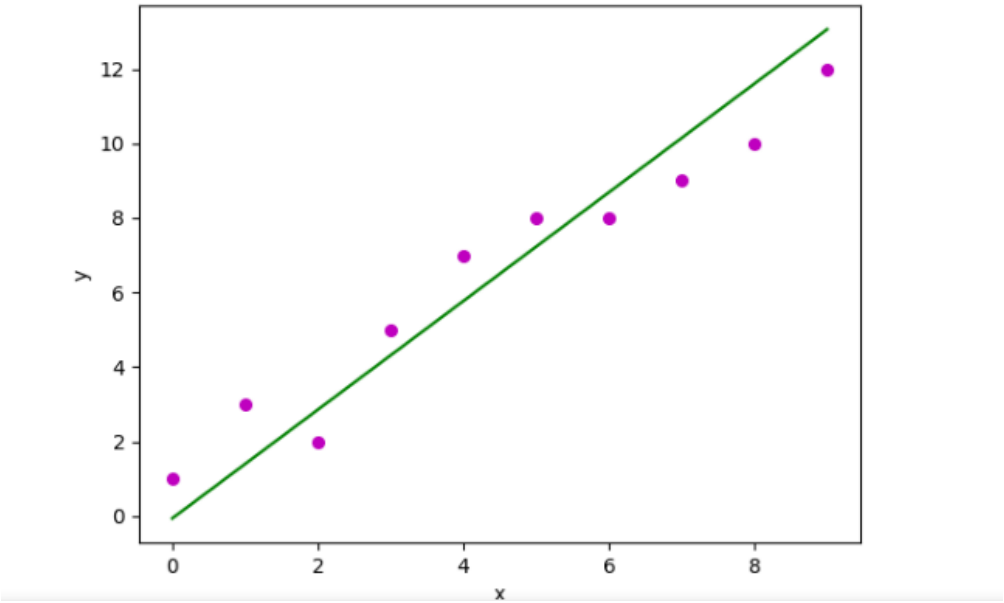
  A scatter plot of the above dataset looks like:



Now, the task is to find a line that fits best in the above scatter plot so that we can predict the response for any new feature values. (i.e a value of x not present in a dataset)

`This line is called a regression line.`

And graph obtained looks like this:



5.1 Linear Regression using `sklearn` - Python Example

Let us see how to use the various Python libraries to implement linear regression on a given dataset.

- We will demonstrate a binary linear model as this will be easier to visualize.
- In this demonstration, the model will use Gradient Descent to learn.

  `Step 1:` Importing all the required libraries

  `Step 2:` Reading the dataset

  `Step 3:` Exploring the data scatter

  `Step 4:` Data cleaning

  `Step 5:` Training our model

  `Step 6:` Exploring our results

Step 1: Importing all the required libraries

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

Step 2: Reading the dataset

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

df = pd.read_csv("Images\\bottles1.csv")

df_binary = df[['Salnty', 'T_degC']]

# Taking only the selected two attributes from the dataset
df_binary.columns = ['Sal', 'Temp']
#display the first 5 rows
df_binary.head()
```

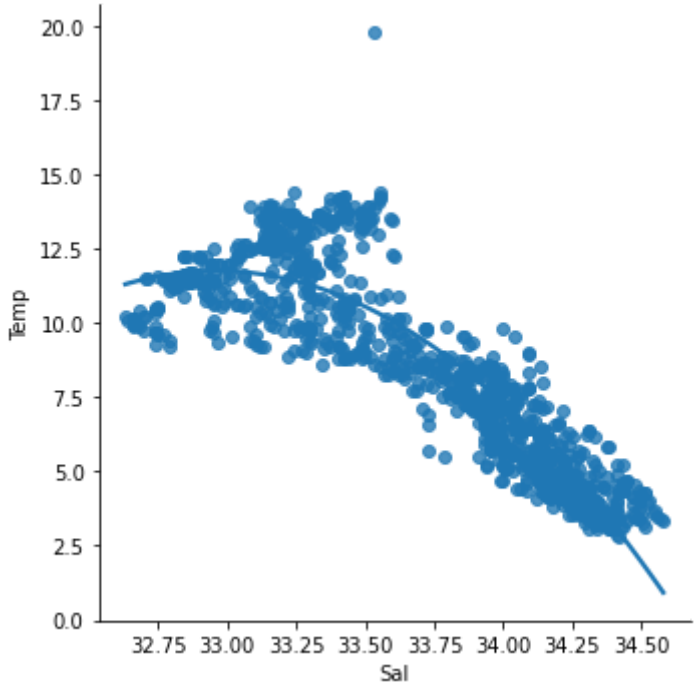
Out[ ]:

	Sal	Temp
0	33.440	10.50
1	33.440	10.46
2	33.437	10.46
3	33.420	10.45
4	33.421	10.45

Step 3: Exploring the data scatter

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

df = pd.read_csv("Images\\bottles1.csv")

df_binary = df[['Salnty', 'T_degC']]

# Taking only the selected two attributes from the dataset
df_binary.columns = ['Sal', 'Temp']
#display the first 5 rows
df_binary.head()

#plotting the Scatter plot to check relationship between Sal and Temp
sns.lmplot(x ="Sal", y ="Temp", data = df_binary, order = 2, ci = None)
plt.show()
```



Step 4: Data cleaning

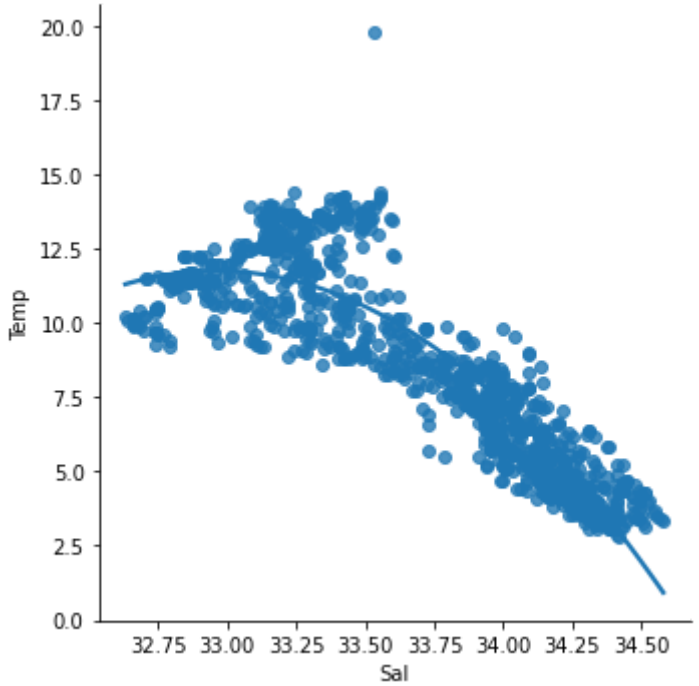
In [ ]:
```
# Eliminating NaN or missing input numbers
df_binary.fillna(method ='ffill', inplace = True)
```

Step 5: Training our model

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

df = pd.read_csv("Images\\bottles1.csv")

df_binary = df[['Salnty', 'T_degC']]
```

```
# Taking only the selected two attributes from the dataset
df_binary.columns = ['Sal', 'Temp']
#display the first 5 rows
df_binary.head()

#plotting the Scatter plot to check relationship between Sal and Temp
sns.lmplot(x ="Sal", y ="Temp", data = df_binary, order = 2, ci = None)

X = np.array(df_binary['Sal']).reshape(-1, 1)
y = np.array(df_binary['Temp']).reshape(-1, 1)

# Separating the data into independent and dependent variables
# Converting each dataframe into a numpy array
# since each dataframe contains only one column
## df_binary.dropna(inplace = True)

# Dropping any rows with Nan values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

# Splitting the data into training and testing data
regr = LinearRegression()

regr.fit(X_train, y_train)
print("Regresssion Score = ",regr.score(X_test, y_test))
```

```
Regresssion Score =  0.7104275483992597
```



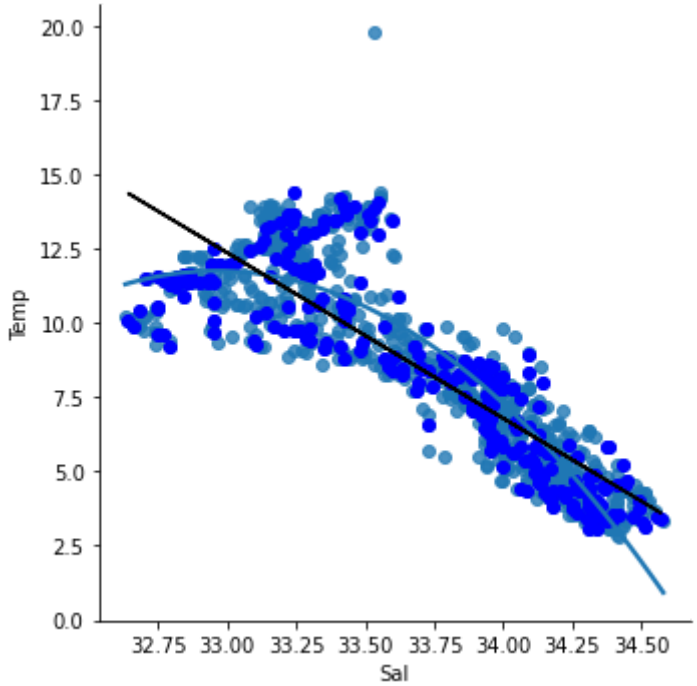
Step 6: Exploring our results

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

df = pd.read_csv("Images\\bottles1.csv")

df_binary = df[['Salnty', 'T_degC']]

# Taking only the selected two attributes from the dataset
df_binary.columns = ['Sal', 'Temp']
#display the first 5 rows
df_binary.head()

#plotting the Scatter plot to check relationship between Sal and Temp
```

```
sns.lmplot(x ="Sal", y ="Temp", data = df_binary, order = 2, ci = None)

X = np.array(df_binary['Sal']).reshape(-1, 1)
y = np.array(df_binary['Temp']).reshape(-1, 1)

# Separating the data into independent and dependent variables
# Converting each dataframe into a numpy array
# since each dataframe contains only one column
## df_binary.dropna(inplace = True)

# Dropping any rows with Nan values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

# Splitting the data into training and testing data
regr = LinearRegression()

regr.fit(X_train, y_train)
print("Regression Score = ",regr.score(X_test, y_test))

y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')

plt.show()
# Data scatter of predicted values
```

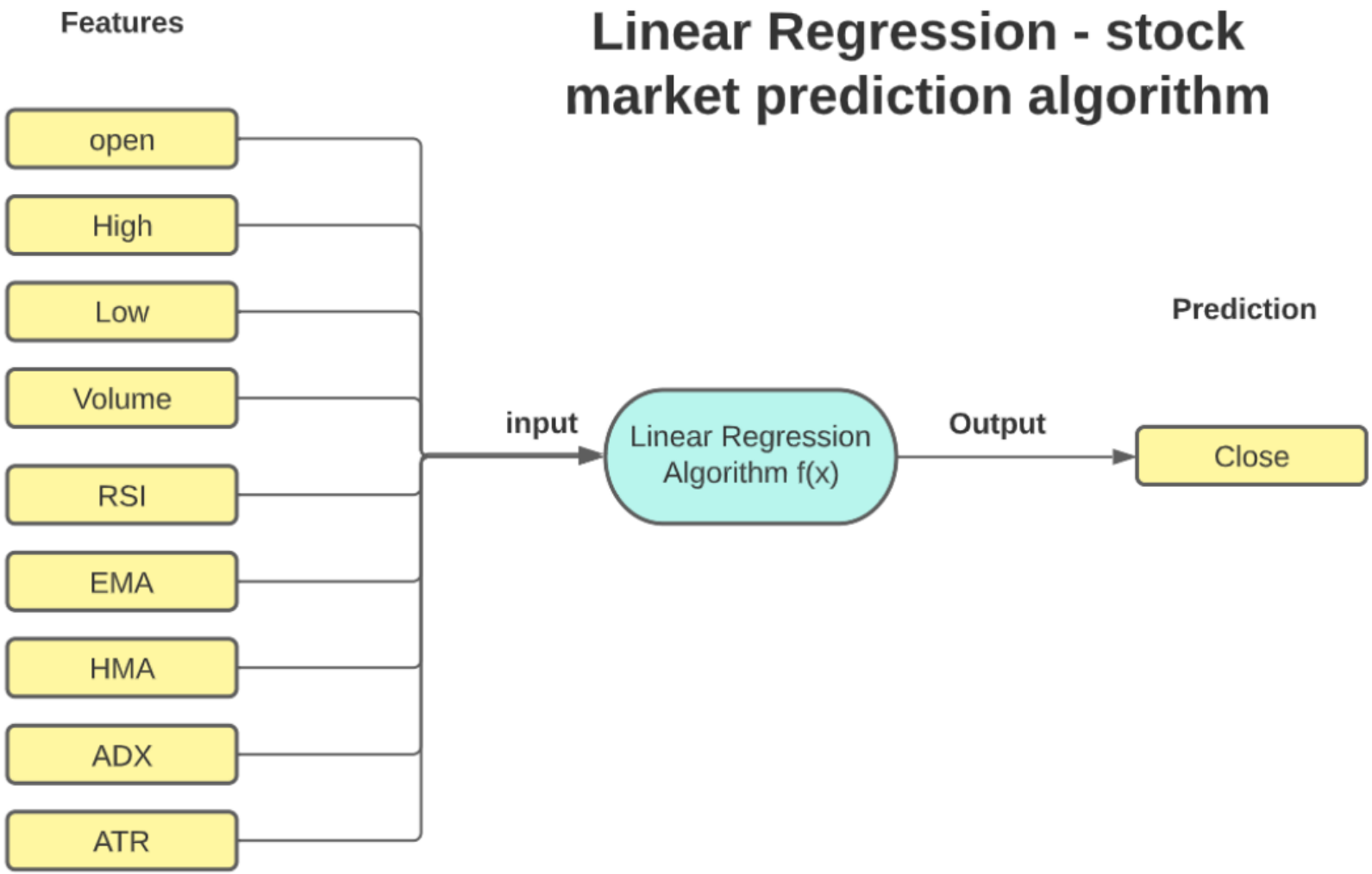
```
Regression Score =  0.7243484524695263
```



5.2 Realtime Linear Regression: Prediction Algorithm for Forecasting Stock Price

- Linear Regression is a widely used algorithm for predicting stock prices.

Preparing the Feature Dataset



In [ ]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.metrics import mean_squared_error, r2_score, explained_variance_score
import numpy as np

# Load the data
df = pd.read_csv('Images\\NIFTY_EOD.csv')

# Prepare the data
X = df[['Open', 'High', 'Low', 'Volume', 'prevclose', 'rsi', 'ema5', 'ema10', 'hma5','hma7','hma9','adx','pdi','mdi','atr']]
y = df['Close']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# Make predictions
y_pred = model.predict(X_test_scaled)

# Compute accuracy metrics
mse = mean_squared_error(y_test, y_pred)
##print("Mean squared error: ", mse)

# Save predicted vs actual values to CSV
df_pred = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df_pred.to_csv('NIFTY_EOD_pred.csv', index=False)



# Compute accuracy metrics
mape = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
r2 = r2_score(y_test, y_pred)
ev = explained_variance_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = np.mean(np.abs(y_test - y_pred))

# Print accuracy metrics
print("PRINT ACCURACY METRICS:")
print("Mean squared error: ", mse)
print("Mean absolute percentage error (MAPE): ", mape)
print("R-squared: ", r2)
```

```
print("Explained variance: ", ev)
print("Mean squared error: ", mse)
print("Root mean squared error (RMSE): ", rmse)
print("Mean absolute error (MAE): ", mae)

# Make a prediction for the next day's close price
last_row = df.tail(1)
last_row_scaled = scaler.transform(last_row[['Open', 'High', 'Low', 'Volume', 'prevclose', 'rsi', 'ema5', 'ema10', 'hma5','hma7','
next_day_pred = model.predict(last_row_scaled)[0]
print("Predicted close price for the next day: ", next_day_pred)
```

```
PRINT ACCURACY METRICS:
Mean squared error:  16.089511432445214
Mean absolute percentage error (MAPE):  0.060458737637536446
R-squared:  0.9999992659810478
Explained variance:  0.9999992660030701
Mean squared error:  16.089511432445214
Root mean squared error (RMSE):  4.011173323660449
Mean absolute error (MAE):  2.268988947333654
Predicted close price for the next day:  17360.236586417002
```

---

6. Multiple Linear Regression

- Multiple linear regression attempts to model the relationship between two or more features and a response by fitting a linear equation to the observed data.
- Clearly, it is nothing but an extension of simple linear regression.
- Consider a dataset with p features(or independent variables) and one response(or dependent variable).
- Also, the dataset contains n rows/observations.

Python implementation of multiple linear regression techniques on the Boston house pricing dataset using Scikit-learn.

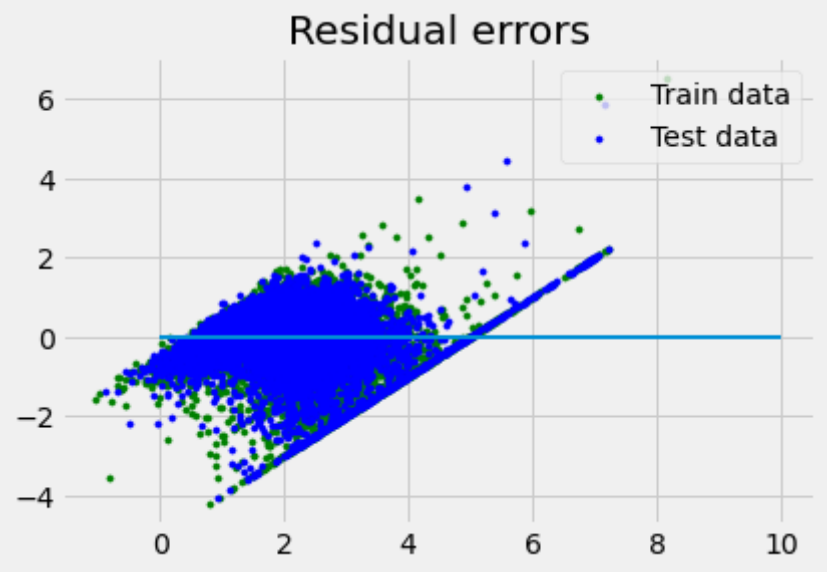
In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
from sklearn.datasets import fetch_california_housing


# load the california dataset
california = fetch_california_housing()
```

```
# defining feature matrix(X) and response vector(y)
X = california.data
y = california.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,
                                                    random_state=1)

# create linear regression object
reg = linear_model.LinearRegression()

# train the model using the training sets
reg.fit(X_train, y_train)

# regression coefficients
print('Coefficients: ', reg.coef_)

# variance score: 1 means perfect prediction
print('Variance score: {}'.format(reg.score(X_test, y_test)))

# plot for residual error

## setting plot style
plt.style.use('fivethirtyeight')

## plotting residual errors in training data
plt.scatter(reg.predict(X_train), reg.predict(X_train) - y_train,
            color = "green", s = 10, label = 'Train data')

## plotting residual errors in test data
plt.scatter(reg.predict(X_test), reg.predict(X_test) - y_test,
            color = "blue", s = 10, label = 'Test data')

## plotting line for zero residual error
plt.hlines(y = 0, xmin = 0, xmax = 10, linewidth = 2)

## plotting legend
plt.legend(loc = 'upper right')

## plot title
plt.title("Residual errors")
```

```
## method call for showing the plot
plt.show()
```

```
Coefficients:  [ 4.36676927e-01  9.58588561e-03 -9.61859974e-02  5.77800722e-01
 -4.33084487e-06 -3.13805195e-03 -4.22347516e-01 -4.34869554e-01]
Variance score: 0.6068373239208641
```

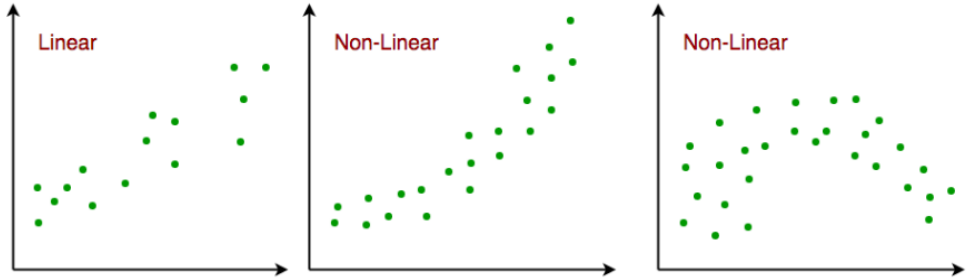


7. Assumptions in Linear Regression Model

- Given below are the basic assumptions that a linear regression model makes regarding a dataset on which it is applied:

Linear relationship:

- Relationship between response and feature variables should be linear.
- The linearity assumption can be tested using scatter plots.
- As shown below, 1st figure represents linearly related variables whereas variables in the 2nd and 3rd figures are most likely non-linear.
- So, 1st figure will give better predictions using linear regression.



Little or no multi-collinearity:

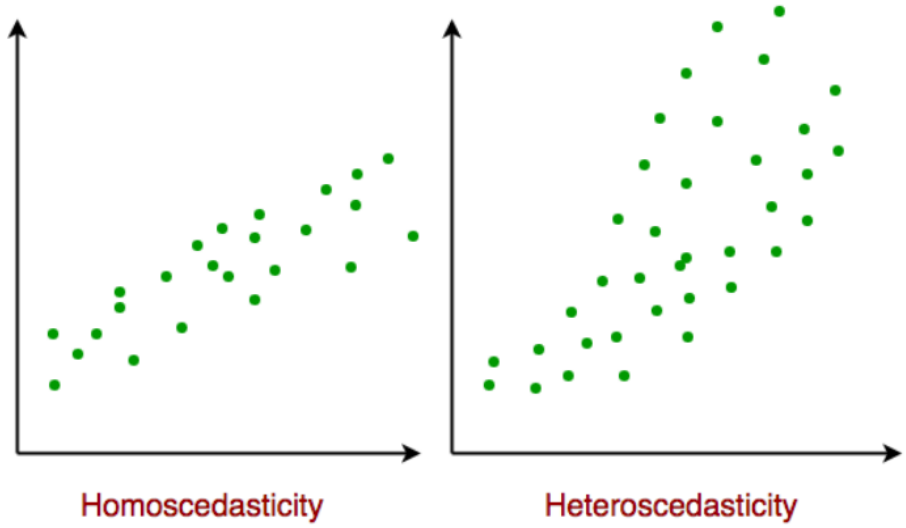
- It is assumed that there is little or no multicollinearity in the data. Multicollinearity occurs when the features (or independent variables) are not independent of each other.

Little or no auto-correlation:

- Another assumption is that there is little or no autocorrelation in the data. Autocorrelation occurs when the residual errors are not independent of each other.

Homoscedasticity:

- Homoscedasticity describes a situation in which the error term (that is, the “noise” or random disturbance in the relationship between the independent variables and the dependent variable) is the same across all values of the independent variables.
- As shown below, figure 1 has homoscedasticity while figure 2 has heteroscedasticity.



Homework Questions

1) Create a code on How do you plot a regression line in Python?

2) Create a code on How do you evaluate the performance of a linear regression model in Python? To evaluate the performance of a linear regression model in Python, you can use metrics such as mean squared error (MSE), mean absolute error (MAE), and R-squared.

3) How do you perform regularized linear regression in Python? To perform regularized linear regression in Python, you can use the Ridge or Lasso regression models from the sklearn.linear_model library.

```
In regularized linear regression, the alpha parameter controls the strength of the regularization. A higher
value of alpha results in stronger regularization and a simpler model.
```

4) Create a sample dataset of 5 input-output pairs, where the input is a single feature (represented by a 1D array) and the output is a single value. We create a LinearRegression object, train it using the sample data, and then use it to make a prediction for a new input value (6). The predicted output value is printed to the console.

5) Multiple Linear Regression - Create a sample dataset of 5 input-output pairs, where the input is a 2D array representing two features and the output is a single value. We create a LinearRegression object, train it using the sample data, and then use it to make a prediction for a new input value

(6 and 12 for the two features). The predicted output value is printed to the console.

6) **Simple Linear Regression - Prediction Creation.**

First generate some sample data x and y using numpy's rand function. We then create a LinearRegression object and fit it to the data using the `fit()` method.

Next, we use the `predict()` method to predict the response for a new value of x, in this case `x=1.`
Finally, we use matplotlib to visualize the fit of the model by plotting the original data points and the predicted values.

7) **Perform multiple linear regression in Python using the scikit-learn library**

- Create a sample dataset with 2 predictor variables (X) and 1 response variable (y).
- We then create a LinearRegression object, fit it to the data, and use it to predict the response for new values of X.
- We also print the coefficients and intercept of the model, and visualize the fit of the model by plotting the true values of y against the predicted values of y using matplotlib.

For solutions of Homework questions, please refer to the `HomeworkSolution.ipynb` file.