Unit-4 Introduction to Machine Learning with Python

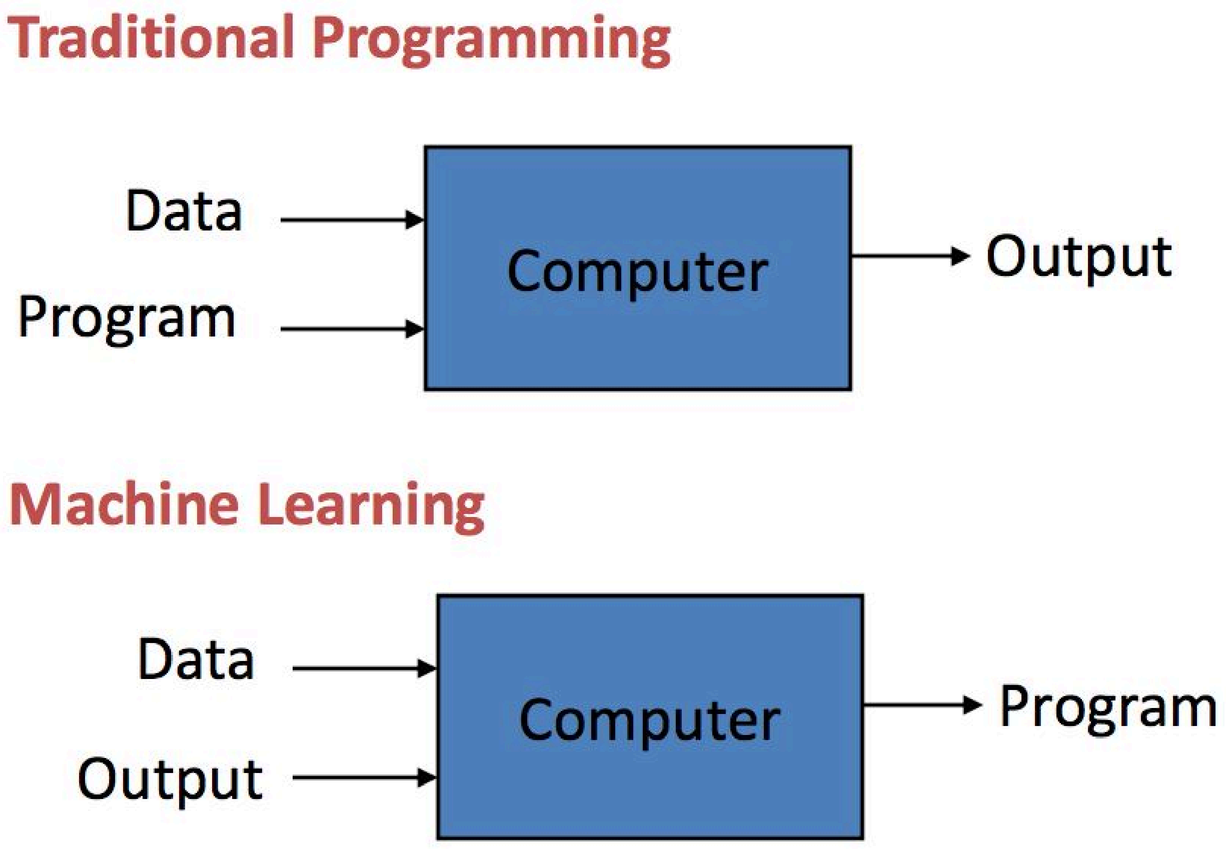
Machine Learning

Machine learning is a field of computer science that uses statical techniques to give computer system the ability to learn with data ,without being explicity programmed

Machine Learning is the science (and art) of programming computers so they can learn from data.

Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.



Types of Machine Learning Systems

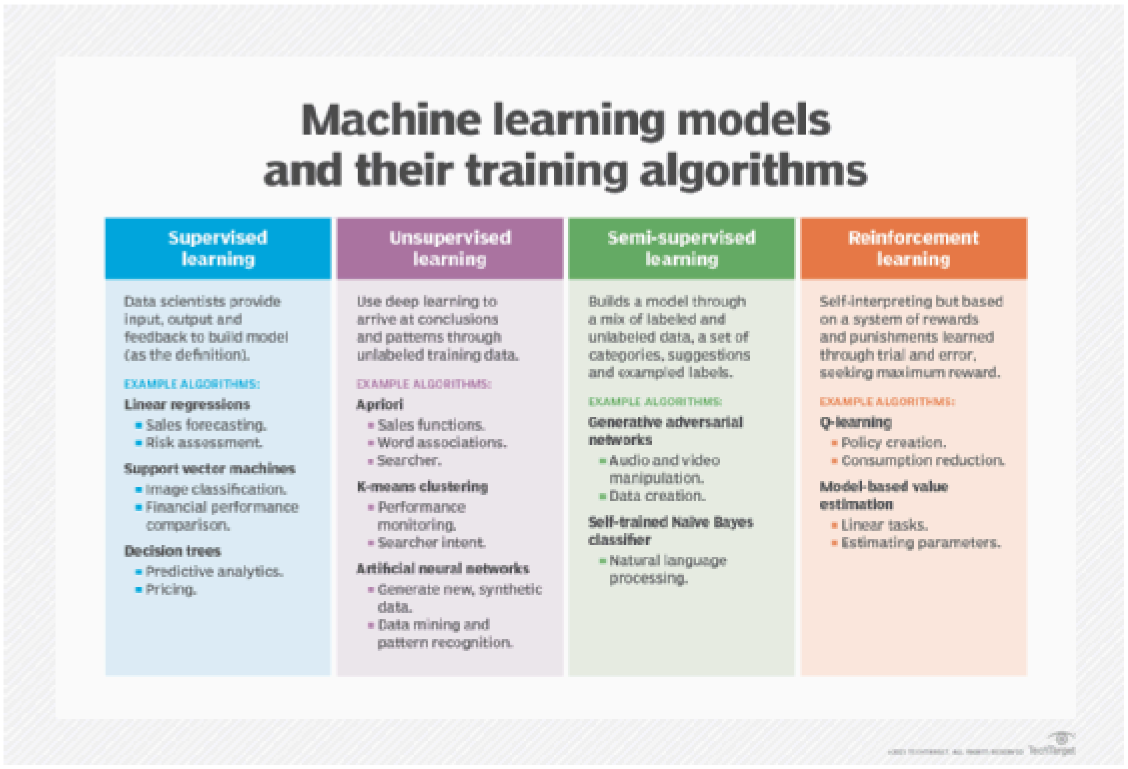
There are so many different types of Machine Learning systems that it is useful to classify them in broad categories based on:

Whether or not they are trained with human supervision (supervised, unsupervised, semisupervised, and Reinforcement Learning)

Whether or not they can learn incrementally on the fly (online versus batch learning) Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model, much like scientists do (instance-based versus model-based learning)

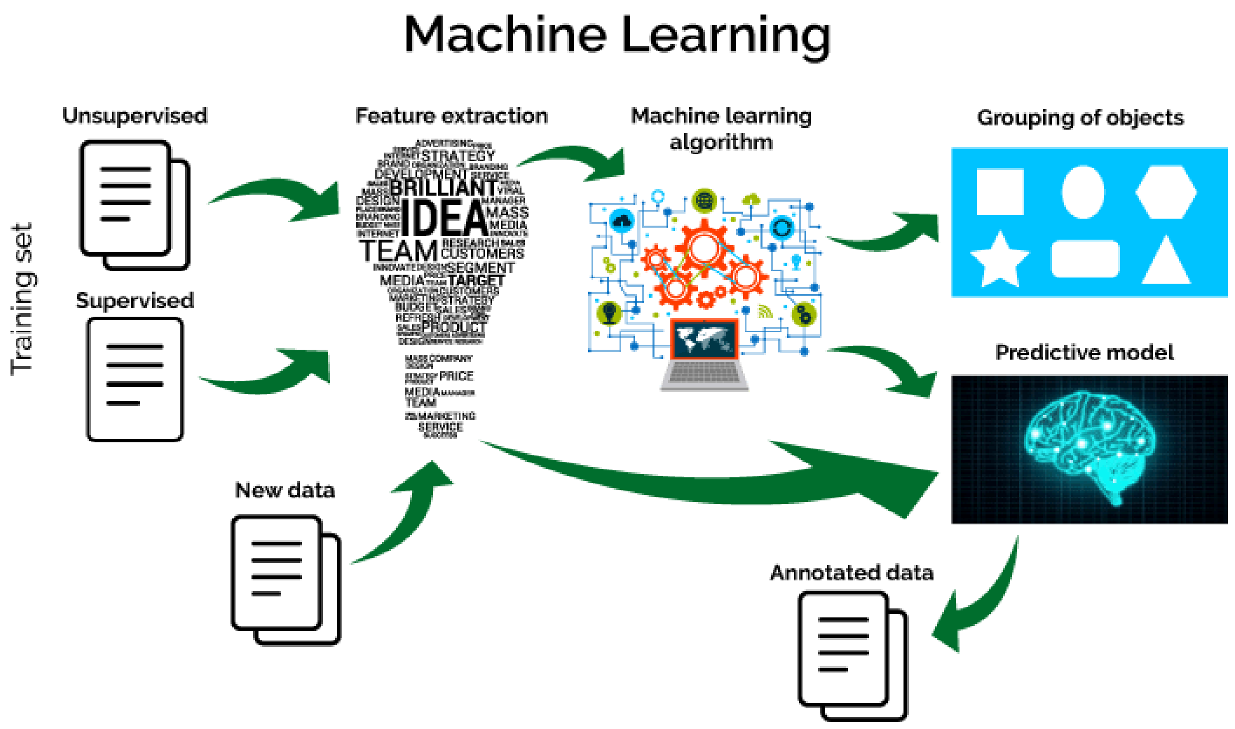
Machine Learning systems can be classified according to the amount and type of supervision they get during training.

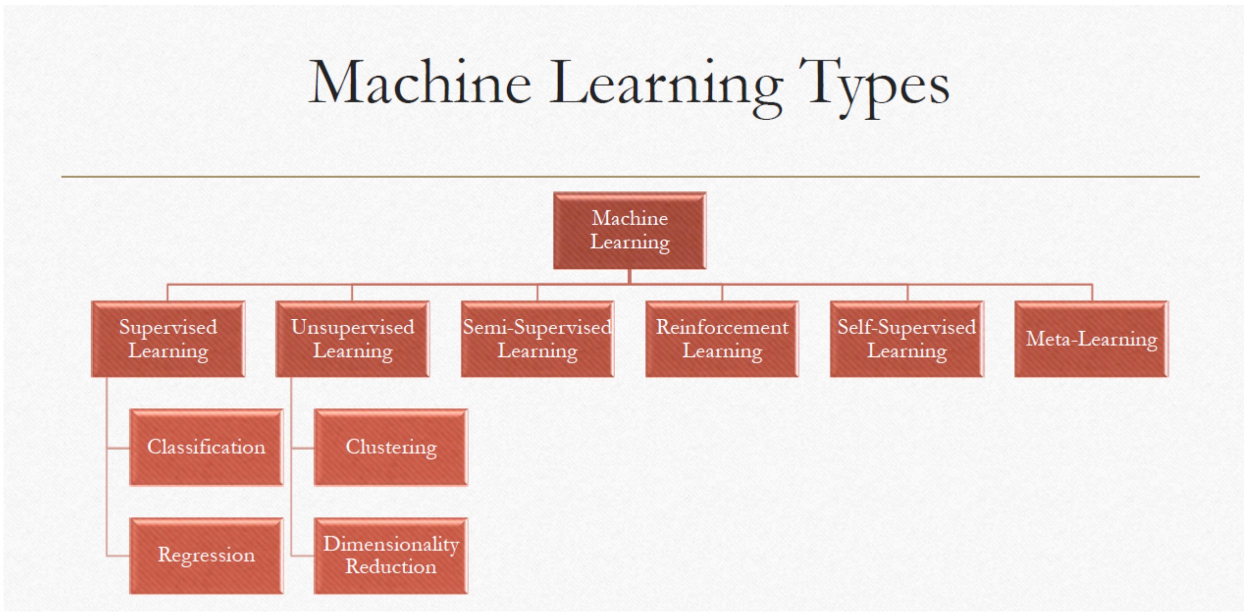
Machine Learning systems can be classified according to the amount and type of supervision they get during training. There are four major categories: supervised learning, unsupervised learning, semisupervised learning, and Reinforcement Learning.



Machine learning models and their training algorithms

Supervised learning	Unsupervised learning	Semi-supervised learning	Reinforcement learning
Data scientists provide input, output and feedback to build model (as the definition).	Use deep learning to arrive at conclusions and patterns through unlabeled training data.	Builds a model through a mix of labeled and unlabeled data, a set of categories, suggestions and exampled labels.	Self-interpreting but based on a system of rewards and punishments learned through trial and error, seeking maximum reward.
EXAMPLE ALGORITHMS:	EXAMPLE ALGORITHMS:	EXAMPLE ALGORITHMS:	EXAMPLE ALGORITHMS:
Linear regressions • Sales forecasting. • Risk assessment. **Support vector machines** • Image classification. • Financial performance comparison. **Decision trees** • Predictive analytics. • Pricing.	**Apriori** • Sales functions. • Word associations. • Searcher. **K-means clustering** • Performance monitoring. • Searcher intent. **Artificial neural networks** • Generate new, synthetic data. • Data mining and pattern recognition.	**Generative adversarial networks** • Audio and video manipulation. • Data creation. **Self-trained Naïve Bayes classifier** • Natural language processing.	**Q-learning** • Policy creation. • Consumption reduction. **Model-based value estimation** • Linear tasks. • Estimating parameters.





Supervised learning

In supervised learning, the training data you feed to the algorithm includes the desired solutions, called labels

some of the most important supervised learning algorithms

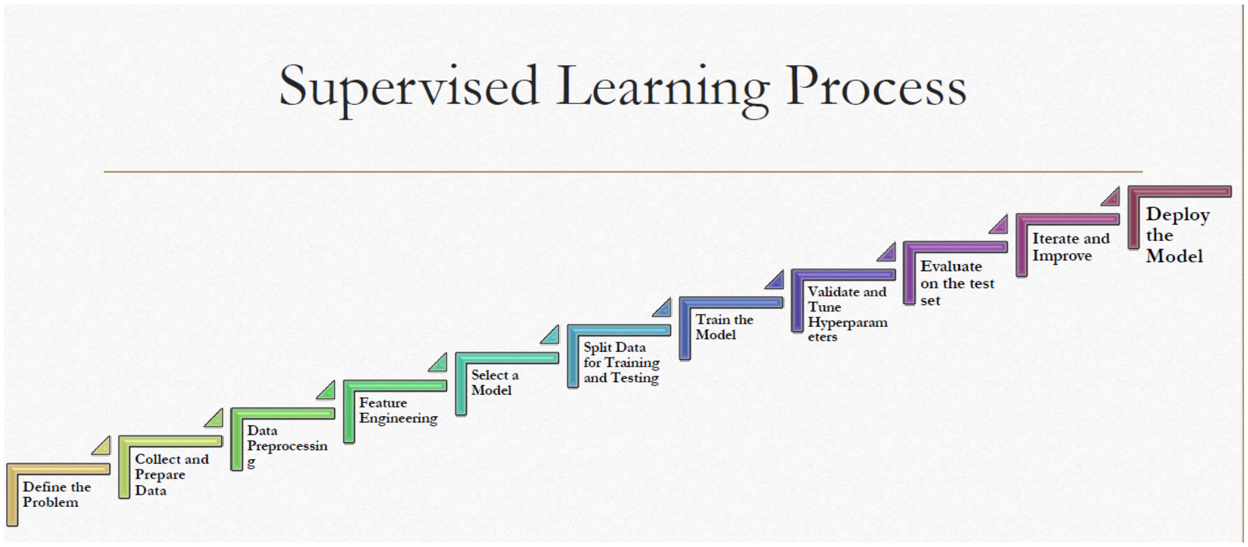
k-Nearest Neighbors

Linear Regression

Logistic Regression

Support Vector Machines (SVMs)

Decision Trees and Random Forests



Unsupervised learning

In unsupervised learning, as you might guess, the training data is unlabeled . The system tries to learn without a teacher. most important unsupervised learning algorithms

• Clustering

K-Means DBSCAN Hierarchical Cluster Analysis (HCA) • Anomaly detection and novelty detection

One-class SVM Isolation Forest • Visualization and dimensionality reduction

Principal Component Analysis (PCA) Kernel PCA Locally-Linear Embedding (LLE) t-distributed Stochastic Neighbor Embedding (t-SNE) • Association rule learning

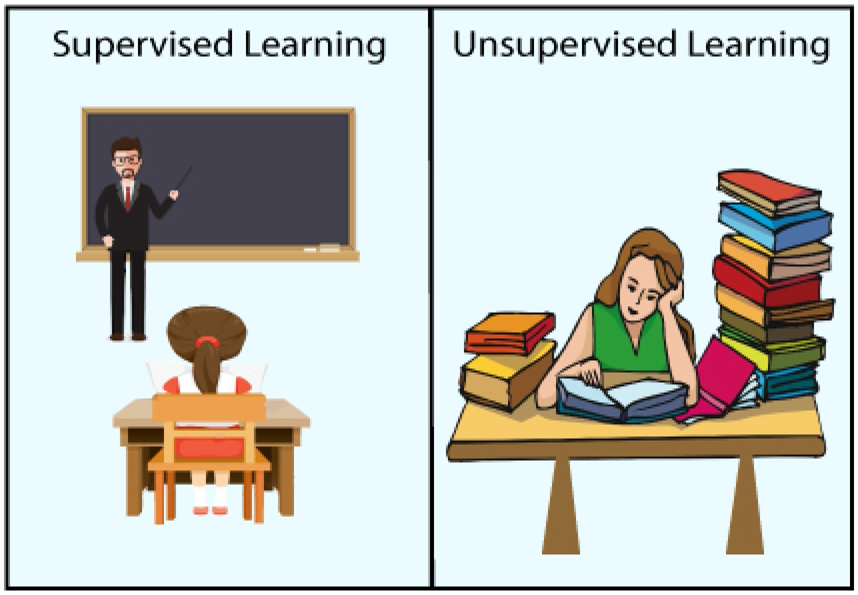
Apriori

Eclat

Some neural network architectures can be unsupervised, such as autoencoders and restricted Boltzmann machines. They can also be semisupervised, such as in deep belief networks and unsupervised pretraining.

Difference between Supervised and Unsupervised Learning

Supervised and Unsupervised learning are the two techniques of machine learning. But both the techniques are used in different scenarios and with different datasets. Below the explanation of both learning methods along with their difference table is given.



Semisupervised learning

Some algorithms can deal with partially labeled training data, usually a lot of unlabeled data and a little bit of labeled data. This is called semisupervised learning Most semisupervised learning algorithms are combinations of unsupervised and supervised algorithms. For example, deep belief networks (DBNs) are based on unsupervised components called restricted Boltzmann machines (RBMs) stacked on top of one another. RBMs are trained sequentially in an unsupervised manner, and then the whole system is fine-tuned using supervised learning techniques.

Reinforcement Learning

Reinforcement Learning is a very different beast. The learning system, called an agent in this context, can observe the environment, select and perform actions, and get rewards in return (or penalties in the form of negative rewards, It must then learn by itself what is the best strategy, called a policy, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.

Regression

Regression searches for relationships among variables. For example, you can observe several employees of some company and try to understand how their salaries depend on their features, such as experience, education level, role, city of employment, and so on.

This is a regression problem where data related to each employee represents one observation. The presumption is that the experience, education, role, and city are the independent features, while the salary depends on them.

In other words, you need to find a function that maps some features or variables to others sufficiently well.

The dependent features are called the dependent variables, outputs, or responses. The independent features are called the independent variables, inputs, regressors, or predictors.

When Do You Need Regression?

Typically, you need regression to answer whether and how some phenomenon influences the other or how several variables are related. For example, you can use it to determine if and to what extent experience or gender impacts salaries.

Regression is also useful when you want to forecast a response using a new set of predictors. For example, you could try to predict electricity consumption of a household for the next hour given the outdoor temperature, time of day, and number of residents in that household.

Regression is used in many different fields, including economics, computer science, and the social sciences. Its importance rises every day with the availability of large amounts of data and increased awareness of the practical value of data.

LINEAR REGRESSION

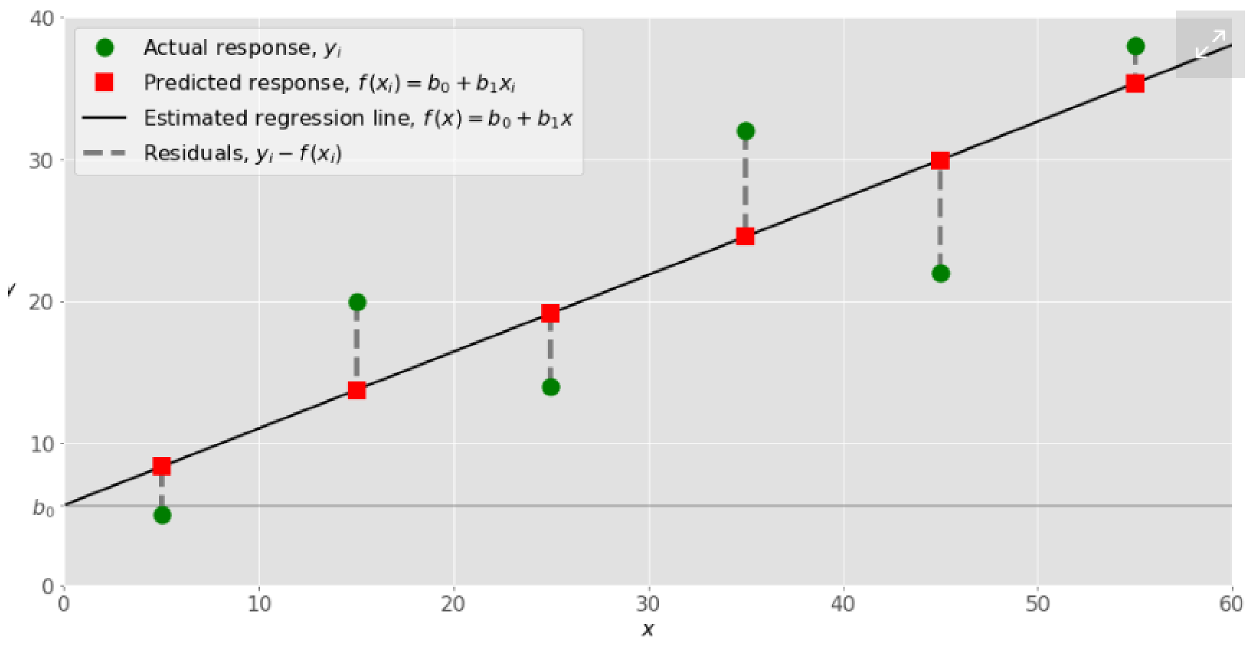
When implementing linear regression of some dependent variable y on the set of independent variables $\mathbf{x} = (x_1, \ldots, x_r)$, where r is the number of predictors, you assume a linear relationship between y and $\mathbf{x}$: $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_r x_r + \varepsilon$. This equation is the regression equation. $\beta_0, \beta_1, \ldots, \beta_r$ are the regression coefficients, and ε is the random error.

Linear regression calculates the estimators of the regression coefficients or simply the predicted weights, denoted with $b_0, b_1, \ldots, b_r$. These estimators define the estimated regression function $f(\mathbf{x}) = b_0 + b_1 x_1 + \cdots + b_r x_r$. This function should capture the dependencies between the inputs and output sufficiently well.

The estimated or predicted response, $f(\mathbf{x}_i)$, for each observation $i = 1, \ldots, n$, should be as close as possible to the corresponding actual response y_i. The differences $y_i - f(\mathbf{x}_i)$ for all observations $i = 1, \ldots, n$, are called the residuals. Regression is about determining the best predicted weights—that is, the weights corresponding to the smallest residuals.

Simple Linear Regression

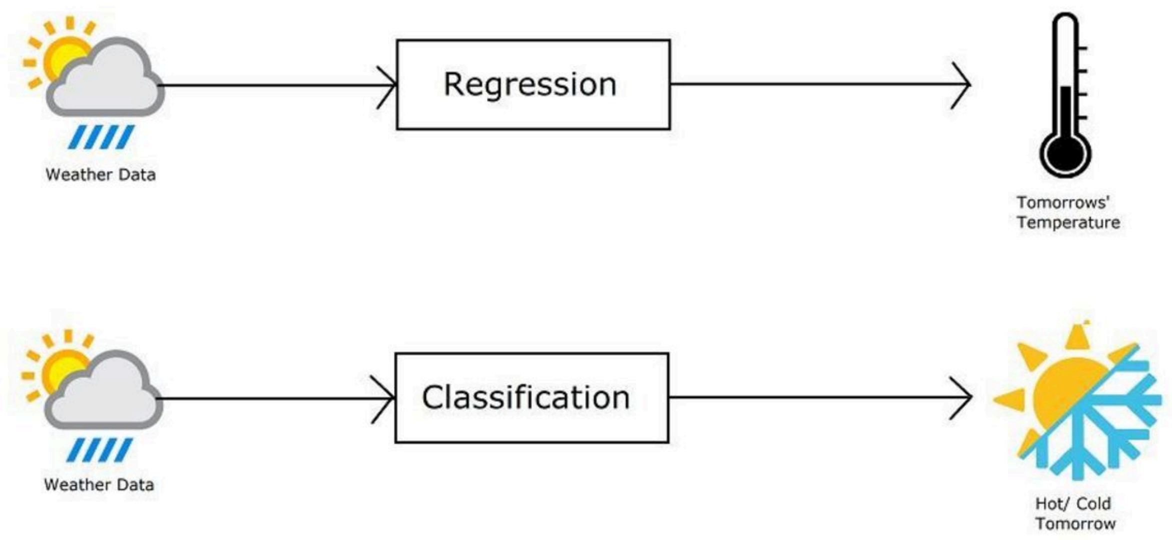
Y=MX+C

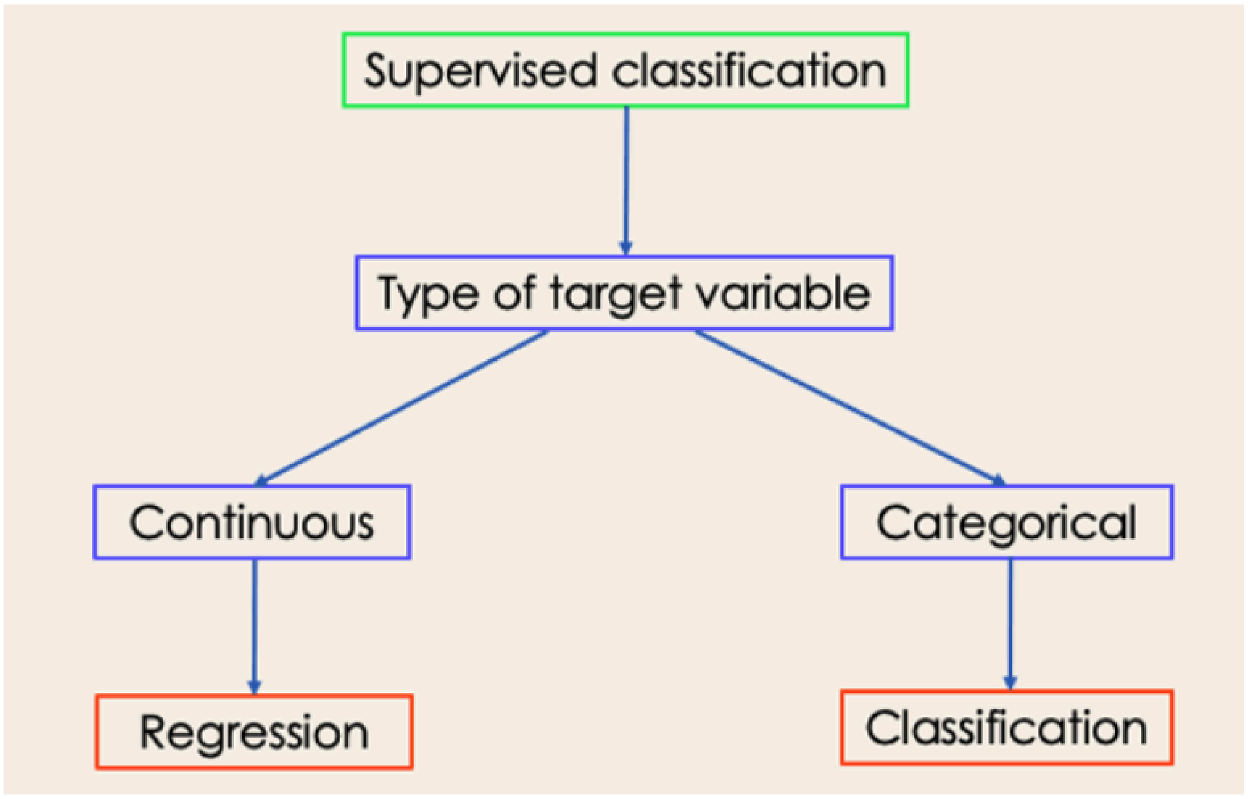


Example of simple linear regression

Regression Vs Classification

Classification and Regression are two major prediction problems that are usually dealt with in Data Mining and Machine Learning.





Regression

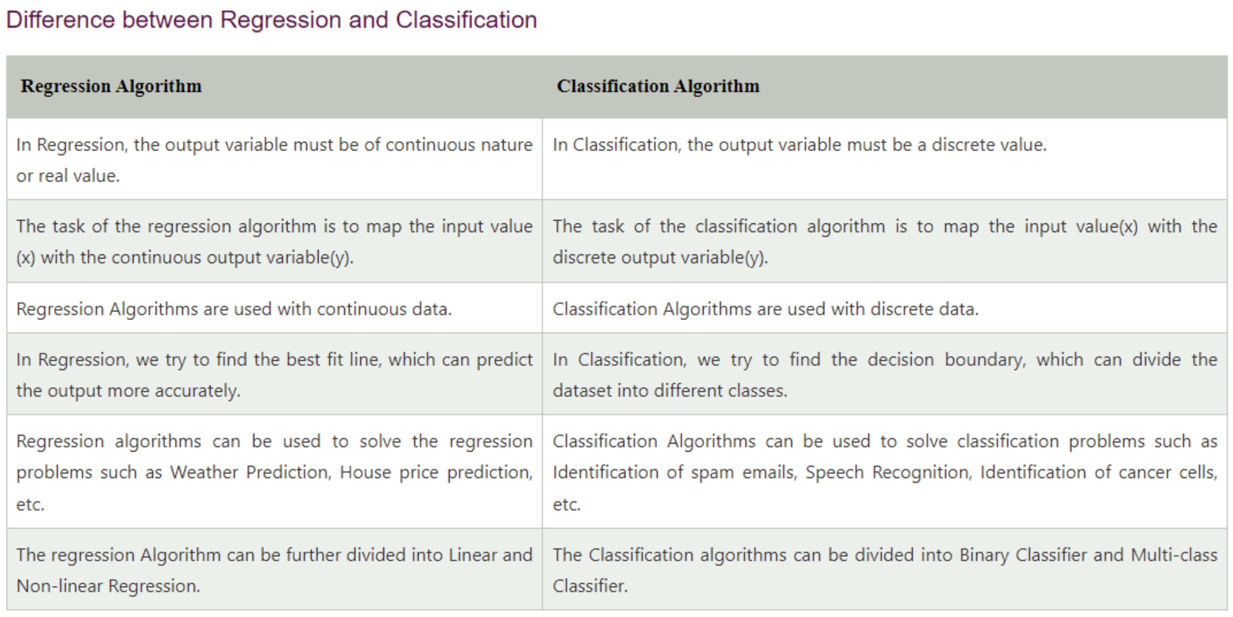
Regression is the process of finding a model or function for distinguishing the data into continuous real values instead of using classes or discrete values. It can also identify the distribution movement depending on the historical data. Because a regression predictive model predicts a quantity, therefore, the skill of the model must be reported as an error in those predictions.

In a regression task, we are supposed to predict a continuous target variable using independent features. In the regression tasks, we are faced with generally two types of problems linear and non-linear regression.

Classification

Classification is the process of finding or discovering a model or function that helps in separating the data into multiple categorical classes i.e. discrete values. In classification, data is categorized under different labels according to some parameters given in the input and then the labels are predicted for the data.

In a classification task, we are supposed to predict discrete target variables(class labels) using independent features. In the classification task, we are supposed to find a decision boundary that can separate the different classes in the target variable.



Difference between Regression and Classification

Regression Algorithm	Classification Algorithm
In Regression, the output variable must be of continuous nature or real value.	In Classification, the output variable must be a discrete value.
The task of the regression algorithm is to map the input value (x) with the continuous output variable(y).	The task of the classification algorithm is to map the input value(x) with the discrete output variable(y).
Regression Algorithms are used with continuous data.	Classification Algorithms are used with discrete data.
In Regression, we try to find the best fit line, which can predict the output more accurately.	In Classification, we try to find the decision boundary, which can divide the dataset into different classes.
Regression algorithms can be used to solve the regression problems such as Weather Prediction, House price prediction, etc.	Classification Algorithms can be used to solve classification problems such as Identification of spam emails, Speech Recognition, Identification of cancer cells, etc.
The regression Algorithm can be further divided into Linear and Non-linear Regression.	The Classification algorithms can be divided into Binary Classifier and Multi-class Classifier.

Data Cleaning

What is Data Cleaning?

In data science and machine learning, the quality of input data is paramount. It's a well-established fact that data quality heavily influences the performance of machine learning models. This makes data cleaning, detecting, and correcting (or removing) corrupt or inaccurate records from a dataset a critical step in the data science pipeline.

Data cleaning is not just about erasing data or filling in missing values. It's a comprehensive process involving various techniques to transform raw data into a format suitable for analysis. These techniques include handling missing values, removing duplicates, data type conversion, and more. Each technique has its specific use case and is applied based on the data's nature and the analysis's requirements.

Common Data Cleaning Techniques

Handling Missing Values:

Missing data can occur for various reasons, such as errors in data collection or transfer. There are several ways to handle missing data, depending on the nature and extent of the missing values.

Deletion:

You remove the instances with missing values from the dataset. While this method is straightforward, it can lead to loss of information, especially if the missing data is not random.

Removing Duplicates:

Duplicate entries can occur for various reasons, such as data entry errors or data merging. These duplicates can skew the data and lead to biased results. Techniques for removing duplicates involve identifying these redundant entries based on key attributes and eliminating them from the dataset.

Data Type Conversion:

Sometimes, the data may be in an inappropriate format for a particular analysis or model. For instance, a numerical attribute may be recorded as a string. In such cases, data type conversion, also known as datacasting, is used to change the data type of a particular attribute or set of attributes. This process involves converting the data into a suitable format that machine learning algorithms can easily process.

Outlier Detection:

Outliers are data points that significantly deviate from other observations. They can be caused by variability in the data or errors. Outlier detection techniques are used to identify these anomalies. These techniques include statistical methods, such as the Z-score or IQR method, and machine learning methods, such as clustering or anomaly detection algorithms.

Data cleaning is a vital step in the data science pipeline. It ensures that the data used for analysis and modeling is accurate, consistent, and reliable, leading to more robust and reliable machine learning models.

```
In [ ]:
```

Data Preprocessing

What is Data Preprocessing?

Data preprocessing is critical in data science, particularly for machine learning applications. It involves preparing and cleaning the dataset to make it more suitable for machine learning algorithms. This process can reduce complexity, prevent overfitting, and improve the model's overall performance.

The data preprocessing phase begins with understanding your dataset's nuances and the data's main issues through Exploratory Data Analysis. Real-world data often presents inconsistencies, typos, missing data, and different scales. You must address these issues to make the data more useful and understandable. This process of cleaning and solving most of the issues in the data is what we call the data preprocessing step.

Skipping the data preprocessing step can affect the performance of your machine learning model and downstream tasks. Most models can't handle missing values, and some are affected by outliers, high dimensionality, and noisy data. By preprocessing the data, you make the dataset more complete and accurate, which is critical for making necessary adjustments in the data before feeding it into your machine learning model.

Data preprocessing techniques include data cleaning, dimensionality reduction, feature engineering, sampling data, transformation, and handling imbalanced data. Each of these techniques has its own set of methods and approaches for handling specific issues in the data

Common Data Preprocessing Techniques

Data Scaling

Data scaling is a technique used to standardize the range of independent variables or features of data. It aims to standardize the data's range of features to prevent any feature from dominating the others, especially when dealing with large datasets. This is a crucial step in data preprocessing, particularly for algorithms sensitive to the range of the data, such as deep learning models.

There are several ways to achieve data scaling, including Min-Max normalization and Standardization. Min-Max normalization scales the data within a fixed range (usually 0 to 1), while Standardization scales data with a mean of 0 and a standard deviation of 1.

Encoding Categorical Variables

Machine learning models require inputs to be numerical. If your data contains categorical data, you must encode them to numerical values before fitting and evaluating a model. This process, known as encoding categorical variables, is a common data preprocessing technique. One common method is One-Hot Encoding, which creates new binary columns for each category/label in the original columns.

Data Splitting

Data Splitting is a technique to divide the dataset into two or three sets, typically training, validation, and test sets. You use the training set to train the model and the validation set to tune the model's parameters. The test set provides an unbiased evaluation of the final model. This technique is essential when dealing with large data, as it ensures the model is not overfitted to a particular subset of data.

Handling Missing Values

Missing data in the dataset can lead to misleading results. Therefore, it's essential to handle missing values appropriately. Techniques for handling missing values include deletion, removing the rows with missing values, and imputation, replacing the missing values with statistical measures like mean, median, or model. This step is crucial in ensuring the quality of data used for training machine learning models.

Feature Selection

Feature selection is a process in machine learning where you automatically select those features in your data that contribute most to the prediction variable or output in which you are interested. Having irrelevant features in your data can decrease the accuracy of many models, especially linear algorithms like linear and logistic regression. This process is particularly important for data scientists working with high-dimensional data, as it reduces overfitting, improves accuracy, and reduces training time.

Three benefits of performing feature selection before modeling your data are:

Reduces Overfitting: Less redundant data means less opportunity to make noise-based decisions.

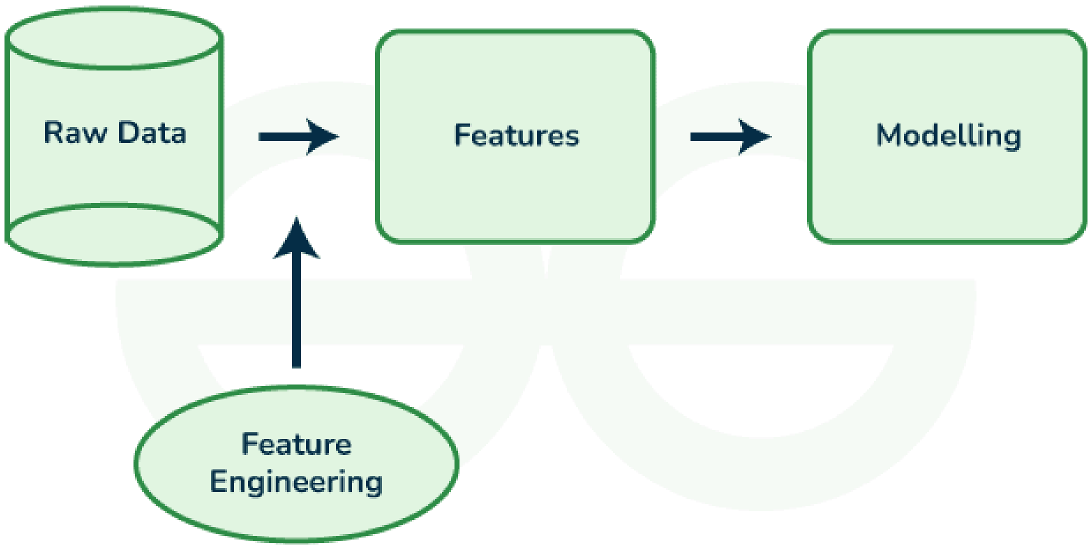
Improves Accuracy: Less misleading data means modeling accuracy improves.

Reduces Training Time: Fewer data points reduce algorithm complexity, and it trains faster

Feature Engineering

Feature engineering is the process of transforming raw data into features that are suitable for machine learning models. In other words, it is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.

The success of machine learning models heavily depends on the quality of the features used to train them. Feature engineering involves a set of techniques that enable us to create new features by combining or transforming the existing ones. These techniques help to highlight the most important patterns and relationships in the data, which in turn helps the machine learning model to learn from the data more effectively.



1. Feature Transformation

Feature transformation is a process in machine learning and data preprocessing where original features are transformed into new features that are more suitable for modeling. This process aims to improve the performance and accuracy of predictive models by making the data more compatible with the chosen algorithms.

Types of Feature Transformation:

Normalization: Rescaling the features to have a similar range, such as between 0 and 1, to prevent some features from dominating others.

Scaling: Scaling is a technique used to transform numerical variables to have a similar scale, so that they can be compared more easily. Rescaling the features to have a similar scale, such as having a standard deviation of 1, to make sure the model considers all features equally.

Encoding: Transforming categorical features into a numerical representation. Examples are one-hot encoding and label encoding.

Transformation: Transforming the features using mathematical operations to change the distribution or scale of the features. Examples are logarithmic, square root, and reciprocal transformations.

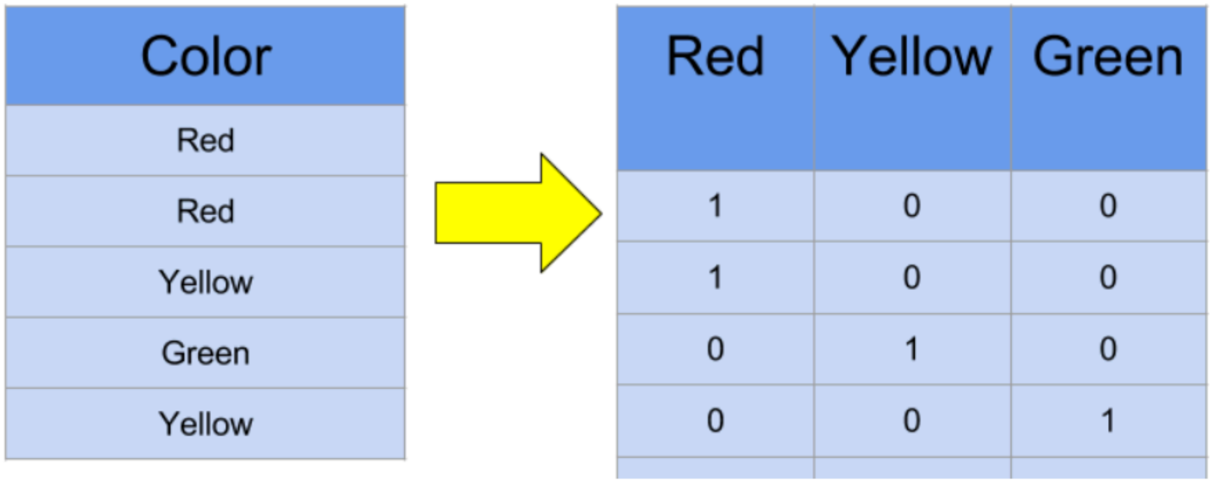
Why Feature Transformation?

Improves Model Performance: By transforming the features into a more suitable representation, the model can learn more meaningful patterns in the data.

Increases Model Robustness: Transforming the features can make the model more robust to outliers and other anomalies.

Improves Computational Efficiency: The transformed features often require fewer computational resources.

Improves Model Interpretability: By transforming the features, it can be easier to understand the model's predictions.



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

2. Feature Selection

Feature Selection is the process of selecting a subset of relevant features from the dataset to be used in a machine-learning model. It is an important step in the feature engineering process as it can have a significant impact on the model's performance.

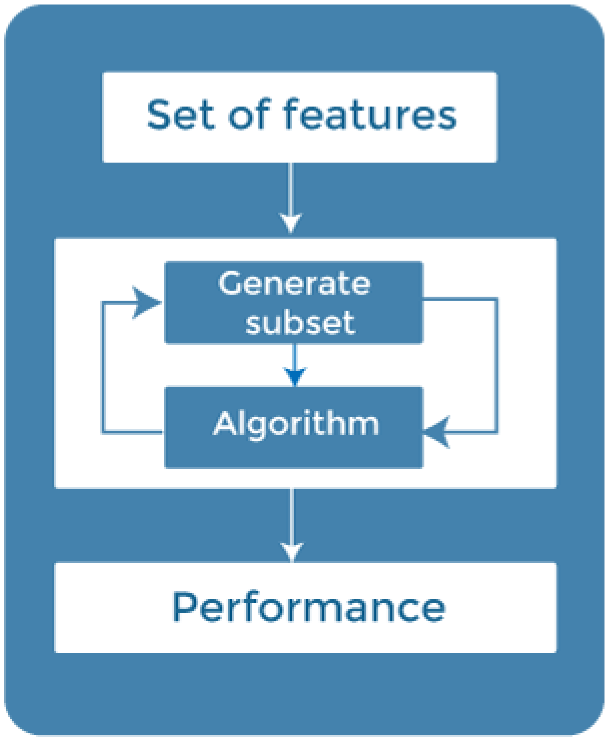
Why Feature Selection?

Reduces Overfitting: By using only the most relevant features, the model can generalize better to new data.

Improves Model Performance: Selecting the right features can improve the accuracy, precision, and recall of the model.

Decreases Computational Costs: A smaller number of features requires less computation and storage resources.

Improves Interpretability: By reducing the number of features, it is easier to understand and interpret the results of the model.



In [14]:
```
import pandas as pd
import numpy as np

# Sample dataset
data = pd.DataFrame({
    'OriginalFeature': [1, 2, 3, 4, 5, 6, 7, 8],
    'CategoricalFeature': ['A', 'B', 'A', 'B', 'A', 'B', 'A', 'C']
})
data
```

Out[14]:

	OriginalFeature	CategoricalFeature
0	1	A
1	2	B
2	3	A
3	4	B
4	5	A
5	6	B
6	7	A
7	8	C

In [15]:
```
# One-Hot Encoding Technique
data1 = pd.get_dummies(data,columns=['CategoricalFeature'])
data1
```

Out[15]:

	OriginalFeature	CategoricalFeature_A	CategoricalFeature_B	CategoricalFeature_C
0	1	1	0	0
1	2	0	1	0
2	3	1	0	0
3	4	0	1	0
4	5	1	0	0
5	6	0	1	0
6	7	1	0	0
7	8	0	0	1

In [16]:
```
# One-Hot Encoding Technique with drop_first=True
data2 = pd.get_dummies(data,columns=['CategoricalFeature'],drop_first=True)
data2
```

Out[16]:

	OriginalFeature	CategoricalFeature_B	CategoricalFeature_C
0	1	0	0
1	2	1	0
2	3	0	0
3	4	1	0
4	5	0	0
5	6	1	0
6	7	0	0
7	8	0	1

In [9]:
```
import pandas as pd
df=pd.read_csv("car data.csv")
df.head()
```

Out[9]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Own
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	

In [10]:
```
df.drop('Car_Name',axis=1,inplace=True) #drop Car_Name Column
```

In [11]:
```
df.head()
```

Out[11]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

One Hot Encoding Technique for Feature Transformation

In [12]:
```python
pd.get_dummies(df) #One-hot Encoding in ML ( Technique for Feature Transformation)
```

Out[12]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner	Fuel_Type_CNG	Fuel_Type_Diesel	Fuel_Type_
0	2014	3.35	5.59	27000	0	0	0	
1	2013	4.75	9.54	43000	0	0	1	
2	2017	7.25	9.85	6900	0	0	0	
3	2011	2.85	4.15	5200	0	0	0	
4	2014	4.60	6.87	42450	0	0	1	
...	...	...	...	...	...	...	...	
296	2016	9.50	11.60	33988	0	0	1	
297	2015	4.00	5.90	60000	0	0	0	
298	2009	3.35	11.00	87934	0	0	0	
299	2017	11.50	12.50	9000	0	0	1	
300	2016	5.30	5.90	5464	0	0	0	

301 rows × 12 columns

In [13]:
```python
df=pd.get_dummies(df,drop_first=True)
df.head() #3 column dropped. we can get conclusion if its not petrol or diesel than it
```

Out[13]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type
0	2014	3.35	5.59	27000	0	0	1	
1	2013	4.75	9.54	43000	0	1	0	
2	2017	7.25	9.85	6900	0	0	1	
3	2011	2.85	4.15	5200	0	0	1	
4	2014	4.60	6.87	42450	0	1	0	

drop_first=True will drop first column after encoding the columns

Adding a New Feature

In [17]:

```
import pandas as pd

#defining features
room_length = [18, 20, 10, 12, 18, 11]
room_breadth = [20, 20, 10, 11, 19, 10]
room_type = ['Big', 'Big', 'Normal', 'Normal', 'Big', 'Normal']

#creating a data frame
data = pd.DataFrame({'Length': room_length, 'Breadth': room_breadth, 'Type': room_type}
data
```

Out[17]:

	Length	Breadth	Type
0	18	20	Big
1	20	20	Big
2	10	10	Normal
3	12	11	Normal
4	18	19	Big
5	11	10	Normal

In [18]:

```
#Adding a feature called area from length and breadth
data['Area'] = data['Length'] * data['Breadth']
data
```

Out[18]:

	Length	Breadth	Type	Area
0	18	20	Big	360
1	20	20	Big	400
2	10	10	Normal	100
3	12	11	Normal	132
4	18	19	Big	342
5	11	10	Normal	110

In [19]:

```
data[data['Area']>300]
```

Out[19]:

	Length	Breadth	Type	Area
0	18	20	Big	360
1	20	20	Big	400
4	18	19	Big	342

Encoding Nominal Variables

In [20]:

```
import pandas as pd
#Creating features
```

```
age = [18, 20, 23, 19, 18, 22]
city = ['City A', 'City B', 'City B', 'City A', 'City C', 'City B']

#Creating a data frame
data1 = pd.DataFrame({'age': age, 'city': city})
data1
```

Out[20]:

```
   age    city
0   18  City A
1   20  City B
2   23  City B
3   19  City A
4   18  City C
5   22  City B
```

In [21]:

```
#get_dummies function of pandas library can be used to dummy code categorical variables
df1=pd.get_dummies(data1)
df1
```

Out[21]:

```
   age  city_City A  city_City B  city_City C
0   18            1            0            0
1   20            0            1            0
2   23            0            1            0
3   19            1            0            0
4   18            0            0            1
5   22            0            1            0
```

In [22]:

```
#drop_first will remove first column of three categories
df2=pd.get_dummies(data1, drop_first=True)
df2
```

Out[22]:

```
   age  city_City B  city_City C
0   18            0            0
1   20            1            0
2   23            1            0
3   19            0            0
4   18            0            1
5   22            1            0
```

Transforming Numeric (continuous) Features to Categorical Features

In [23]:
```
import pandas as pd
#Defining features
apartment_area = [4720, 2430, 4368, 3969, 6142, 7912]
apartment_price = [2360000,1215000,2184000,1984500,3071000,3956000]

#Creating a data frame
data4 = pd.DataFrame({'Area':apartment_area, 'Price': apartment_price})
data4
```

Out[23]:

	Area	Price
0	4720	2360000
1	2430	1215000
2	4368	2184000
3	3969	1984500
4	6142	3071000
5	7912	3956000

In [24]:
```
import numpy as np
data4['Price'] = np.where(data4['Price'] > 3000000, 'High', np.where(data4['Price'] < 2
data4
```

Out[24]:

	Area	Price
0	4720	Medium
1	2430	Low
2	4368	Medium
3	3969	Low
4	6142	High
5	7912	High

In [25]:
```
#Understanding the use of where function of numpy
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
x = np.where(arr%2 == 1,'Odd','Even')
print(arr)
print(x)
```

```
[1 2 3 4 5 6 7 8]
['Odd' 'Even' 'Odd' 'Even' 'Odd' 'Even' 'Odd' 'Even']
```

In [27]:
```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9.5])
arr = np.where(arr%2 == 1,'Odd',np.where(arr%2==0,'Even','medium'))
print(arr)
```

```
['Odd' 'Even' 'Odd' 'Even' 'Odd' 'Even' 'Odd' 'Even' 'medium']
```

```
In [ ]:
```