Chapter - 1: Prerequisites

Basics of Machine Learning

Machine learning is a set of techniques to make computers better at doing things that humans traditionally can do better than machines. In other words, machine learning comes in hand in those cases, where humans outperform machines. For example, driving cars (tesla, uber are trying to focus on that matter, but not used in all vehicles), natural language conversation etc.

Machine Learning (ML) is a branch of artificial intelligence (AI) that allows systems to learn from data, identify patterns, and make decisions with minimal human intervention. Instead of being explicitly programmed to perform tasks, ML algorithms use statistical techniques to improve their performance as they are exposed to more data.

Key Concept: In ML, the system learns from training data, which contains input-output pairs. The algorithm then tries to generalize from the data to make accurate predictions or decisions on new, unseen data.

Example of Machine Learning:

Spam Email Detection:

Training Data: A large set of emails labeled as either "spam" or "not spam."

Algorithm: A machine learning model (like a decision tree or neural network) is trained on these labeled examples. It learns patterns that distinguish spam emails from regular ones, such as specific keywords, sender information, or formatting.

Prediction: Once trained, the model can predict whether new, unseen emails are spam or not based on the patterns it learned during training. Over time, as it processes more emails and learns from its mistakes (incorrect classifications), the system can improve its accuracy.

Types of Machine Learning:

1. Supervised Learning: The model learns from labeled data (e.g., spam vs. not spam).
2. Unsupervised Learning: The model finds patterns in unlabeled data (e.g., customer segmentation).
3. Reinforcement Learning: The model learns by interacting with an environment and receiving feedback (rewards or penalties).

Comparison of ML techniques

Aspect	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Definition	The model learns from labeled data (input-output pairs)	The model learned from unlabeled data to find inherited patterns	The model learns by interactive with an environment, receiving rewards or penalties
Goal	Predict or classify outputs based on labeled inputs	Find hidden patterns, structures or groupings in the data	Maximize cumulative reward by learning actions through trial and error
Data Type	Labeled data points (contains input-output pairs)	Unlabeled data (only inputs are provided).	Feedback-based (rewards/penalties) without labeled input-output pairs.
Example	Predicting house prices based on features (size, location).	Clustering customers based on purchasing behavior.	Training a robot to navigate a maze.
Common Algorithms	Decision Trees, Support Vector Machines, Neural Networks	K-means, Hierarchical Clustering, PCA	Q-Learning, Deep Q-Network (DQN), Policy Gradients

Training Approach	Uses historical labeled data to learn.	Find patterns or groupings without labels.	Learn through trial and error from the environment.
Feedback System	Direct feedback (correct label is given).	No explicit feedback, only patterns to discover.	Feedback is in the form of rewards or penalties after actions.
Application Areas	Email classification (spam vs. not spam), fraud detection.	Market segmentation, anomaly detection.	Self-driving cars, game-playing AI (like AlphaGo).

Real life application of ML

Some real life applications of ML algorithm is given as -

1.**Email Spam Detection:**

- Type of ML: Supervised Learning
- Explanation: The system is trained on labeled emails (spam vs. not spam) and learns to classify new emails as spam or not.

2.**Customer Segmentation for Marketing:**

- Type of ML: Unsupervised Learning
- Explanation: ML is used to segment customers into groups based on behavior or purchasing patterns, without any predefined labels.

3.**Product Recommendation (e.g., Amazon, Netflix):**

- Type of ML: Unsupervised Learning and Supervised Learning
- Explanation: Recommendation systems analyze user behavior to suggest relevant products. Collaborative filtering is often unsupervised, while classification models can be supervised.

4.**Autonomous Vehicles (Self-driving Cars):**

- Type of ML: Reinforcement Learning
- Explanation: The car learns to navigate roads by interacting with the environment, receiving feedback in the form of rewards (staying on course) and penalties (collisions).

5.**Fraud Detection in Banking:**

- Type of ML: Supervised Learning
- Explanation: ML models are trained on historical labeled transactions (fraudulent vs. non-fraudulent) and learn to detect suspicious activities in real-time.

6.**Medical Diagnosis:**

- Type of ML: Supervised Learning
- Explanation: Algorithms are trained using labeled patient data (e.g., symptoms and corresponding diagnoses) to predict diseases or conditions for new patients.

7.**Customer Churn Prediction:**

- Type of ML: Supervised Learning
- Explanation: Predictive models are used to forecast which customers are likely to leave a service, based on historical data of past customer behaviors.

8. **Anomaly Detection in Network Security:**

- Type of ML: Unsupervised Learning
- Explanation: In cybersecurity, ML models can identify unusual patterns in network traffic (anomalies) without pre-labeled data to prevent attacks.

9.**Virtual Personal Assistants (e.g., Siri, Alexa):**

- Type of ML: Supervised Learning and Reinforcement Learning

- Explanation: These assistants use supervised learning for understanding speech (speech-to-text) and reinforcement learning to improve their responses based on user interaction.

10. **Stock Market Prediction:**
    - Type of ML: Supervised Learning
    - Explanation: Historical stock data with labeled trends (e.g., increase or decrease) are used to train models that predict future stock movements.

These applications span various industries, showcasing how ML techniques like supervised, unsupervised, and reinforcement learning can solve diverse problems based on the available data and desired outcomes.

Importance of ML in current situation

Machine Learning (ML) is crucial in the current world due to its ability to handle and derive meaningful insights from vast amounts of data, automate decision-making processes, and enhance efficiency across industries. Here's why ML is so important today:

1. **Data Explosion and Big Data:** With the rapid growth of digital platforms, IoT devices, social media, and online transactions, enormous amounts of data are being generated every second. ML is essential because it can process, analyze, and extract valuable insights from this data that are beyond human capacity to manage manually. Companies use ML to make data-driven decisions and predict trends.
2. **Automation of Repetitive Tasks:** ML enables the automation of routine, repetitive tasks that were traditionally done by humans, reducing errors and increasing efficiency. For example, chatbots can handle customer service inquiries, while ML algorithms can process documents or classify emails automatically. This automation reduces costs and frees up human resources for more complex tasks.
3. **Improvement in Decision Making:** ML helps organizations make more informed and precise decisions by analyzing large datasets for trends, patterns, and correlations. It can provide real-time predictive insights, like forecasting demand, identifying risks, and making pricing decisions, which allows businesses to be more proactive and competitive.
4. **Personalization:** One of the biggest advantages of ML is the ability to provide personalized experiences. From product recommendations on e-commerce websites (like Amazon) to personalized content on social media (like Instagram), ML helps tailor the user experience. This increases user engagement, satisfaction, and sales.
5. **Enhanced Customer Experience:** ML enhances customer experience by enabling real-time support through virtual assistants, chatbots, and recommendation engines. Companies like Netflix, Spotify, and Amazon use ML to provide personalized content suggestions that improve customer retention and satisfaction.
6. **Improved Healthcare:** In healthcare, ML is revolutionizing diagnostics, treatment plans, and drug discovery. ML models can analyze patient data to detect diseases earlier, recommend personalized treatments, and even assist in medical imaging to identify conditions like cancer more accurately and efficiently.
7. **Financial and Business Efficiency:** In finance, ML is used for fraud detection, credit scoring, algorithmic trading, and risk management. Businesses use ML for demand forecasting, supply chain optimization, and customer segmentation, driving higher operational efficiency.
8. **Enabling New Technologies (AI, Robotics, Autonomous Systems):** ML is a key driver behind innovations such as AI, robotics, and autonomous systems. Self-driving cars, drones, and smart robots rely on reinforcement learning and other ML techniques to navigate environments, make decisions, and improve their performance over time without human intervention.
9. **Cybersecurity and Fraud Prevention:** ML is critical in cybersecurity, as it helps detect unusual patterns in network traffic and identify potential cyber threats in real-time. Banks and financial institutions use ML models to detect fraudulent transactions by analyzing the behavior of users and flagging anomalies.

10. **Scalability and Adaptability:** ML models can scale easily as more data becomes available, continuously improving their accuracy and relevance. Unlike traditional systems that require manual updates, ML models adapt automatically as they are exposed to new information, making them highly versatile across industries.

Chapter - 2: Linear Regression

Concept of Linear Regression

The concept of linear regression (Supervised Algorithm) is derived from the below questions -

1. **Input** and **output** variables
2. Is there a relationship between **input** and **output** variables?
3. How strong is the relationship? ***Accurate prediction***
4. Can we **quantify** the effect of the relationship?
5. Is the relationship approximately linear? If yes, then ***Linear Regression***

Output and Input variables in Linear Regression

Output variables: The variable that is to be predicted. For linear regression tasks (for simple and multiple linear regression), the output variable must be continuous in nature. The other names of output variables are dependent variable, outcomes, response variable, target variables etc. For example, the price of a book in rupees, sales figure in a month in dollars, BMI of a person etc.

Input variables: The variables that are provided by the user to make the linear regression model. The input variables may be continuous or categorical in nature. For categorical variables, one-hot encoded form is used. The other names of input variables are independent variables, features, predictors, covariates etc. For example, popularity of a book, marketing value of a product, height of a person etc.

Population Model

In general, probabilistic models can be used for understanding the population. For example, in the 'MPG_Car_Data.csv' data set fuel efficiency (mpg) can be the response variable (output) and the horsepower (hp) can be the independent variable (output). For a random car, suppose the output variable is denoted by Y and the input variable is denoted by X. There is a probability distribution of X in the population and Y has a conditional probability distribution given X. So, considering all facts, a typical non-linear model can be given as: $Y = f(X) + \varepsilon$ for an unknown nonlinear function f, where ε is a random error term. For example, for the 'MPG_Car_Data.csv' data set, we can write that $Y = \frac{1.8}{X} - 0.03X + \varepsilon$

Mathematical Model for Linear Regression

In general, the input variables are represented as vector X, whose elements are $\{x_1, x_2, \ldots., x_n\}$. The output variable is represented as Y. A typical linear regression is represented as -

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \varepsilon \qquad \textbf{for multiple linear regression model}$$

$$Y = \beta_0 + \beta_1 x_1 + \varepsilon \qquad \textbf{for simple linear regression model}$$

When there are more than one input variables, the regression model is called a multiple linear regression model, otherwise it is a simple linear regression model. The coefficient β_0 indicates the intercept and the other coefficients $\beta_i \, for \; i = 1,2,\ldots,n$ indicates the slopes of features.

Assumptions in Simple Linear Regression Model (SLRM)

There are mainly 4 assumptions in a SLRM. They are -

1. For a random ith sample, note that in the linear approximation $Y^{(i)} = \beta_0 + \beta_1 {x_1}^{(i)} + \varepsilon^{(i)}$, the random error term $\varepsilon^{(i)}$ for the ith sample is the same as its residual $R^{(i)}$
2. For deriving the ordinary least squares estimates, $\widehat{\beta_0}$ and $\widehat{\beta}_1$, no assumptions about $\varepsilon^{(i)}$ are needed.

3. For the purpose of deriving statistical inferences (mean, variance) about the least square estimates, it is assumed that $\varepsilon^{(i)}$ will have zero mean, constant variance and uncorrelated across the samples that will be chosen from the population.
4. For the purpose of constructing hypothesis tests and confidence intervals for the least squares estimates, we will also assume that $\varepsilon^{(i)}$ is normally distributed.

SLRM Python Notebook

In this notebook, the 'MPG_Car_Data.csv' file is used. The 'horsepower' column is treated as the input variable (feature) and the 'mpg' column is treated as the output variable (dependent variable). A SLRM model 'slrm_model' is created by using Python's sklearn module's LinearRegression() object, which comes under the linear_model library. Different parameters and methods are covered in this notebook, including how to build SLRM models.

MLRM Python Notebook

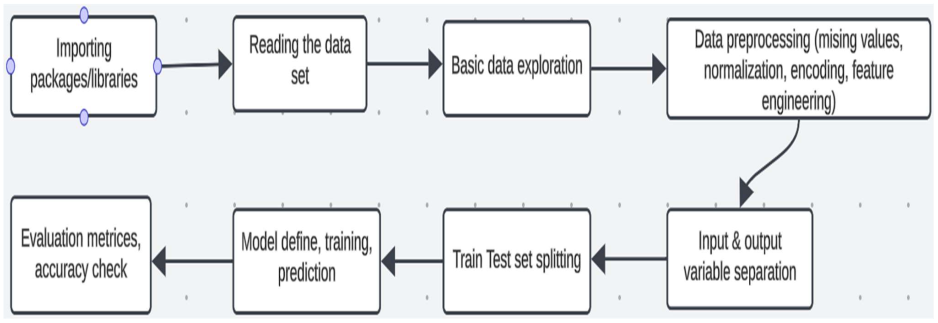
In this notebook, the 'MPG_Car_Data.csv' file is used. The 'mpg' column is treated as output of the 'mlrm_model' model. Whereas, the columns 'cylinders', 'weight', 'displacement', 'acceleration' and 'horsepower' are treated as input variables or features. The model is created by using Python's sklearn module's LinearRegression() object, which comes under the linear_model library. Different parameters and methods are covered in this notebook, including how to build MLRM models.

Linear Regression using Categorical Variables

In this notebook, a MLRM is covered with mixed types of variables in very short. The column 'mpg' is considered as output and the columns 'acceleration', 'horsepower' and 'cylinders' are considered as inputs. Besides that, the categorical form of 'displacement' column discat and that of 'weight' column wthat are considered as inputs. A basic linear regression model is built in this notebook that is provided.

A complete Linear Regression Model in Python

In this Python notebook, a detailed method is provided that deals with building linear regression machine learning models. The data set under consideration is called 'Saratoga_House_Price.csv' and it is a mix of continuous and categorical variables. The 'price' column is the target variable. This notebook contains data exploration, train-test split using sklearn, model building, checking accuracy (evaluation metrics) and some basic data visualization methods.


Importing packages/libraries
Reading the data set
Basic data exploration
Data preprocessing (mising values, normalization, encoding, feature engineering)
Input & output variable separation
Train Test set splitting
Model define, training, prediction
Evaluation metrices, accuracy check

Chapter - 3: Gradient Descent and Cost Function

Basics of Gradient Descent

Gradient Descent is an optimization algorithm commonly used in machine learning to minimize the cost function or loss function of a model. It works by iteratively adjusting the model's parameters (such as weights in a neural network) in order to find the values that result in the lowest possible error between the model's predictions and the actual outcomes.

Key Concepts:

1. **Cost Function (Loss Function):** A function that measures how well the model is performing. It quantifies the difference between predicted values and actual values. Gradient descent aims to minimize this function.
2. **Gradient:** The gradient is the vector of partial derivatives of the cost function with respect to the model's parameters. It points in the direction of the steepest increase of the cost function. To minimize the cost function, we move in the opposite direction of the gradient.
3. **Learning Rate (α)**: A hyperparameter that controls the step size in the gradient descent process. It dictates how much to change the model's parameters in response to the gradient. A large learning rate may overshoot the minimum, while a small learning rate can result in slow convergence.

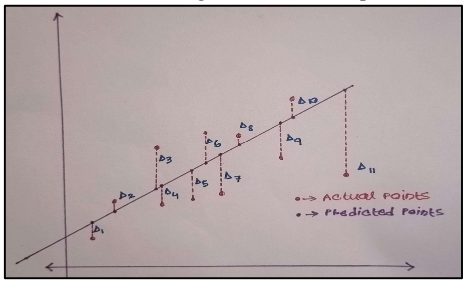
Gradient Descent Process:

At each iteration, the algorithm adjusts the parameters θ by taking a step in the direction opposite to the gradient of the cost function. Mathematically, the update rule for the parameters is: $\theta := \theta - \alpha . \nabla j(\theta)$; Where θ's are the model parameters, α is the learning rate,
$\nabla j(\theta)$ is the gradient of the cost function $j(\theta)$.

Types of Gradient Descent:

1. Batch Gradient Descent: Uses the entire dataset to compute the gradient in each step. This can be computationally expensive for large datasets.
2. Stochastic Gradient Descent (SGD): Updates the parameters using the gradient calculated from a single data point (or a small subset). It is faster but introduces more noise in the updates.
3. Mini-Batch Gradient Descent: A compromise between batch and stochastic gradient descent. It updates parameters based on a small batch of data, reducing the variance while being faster than full batch.

Visualization: Imagine gradient descent as trying to find the lowest point in a valley (minimizing the cost function) by taking steps downhill (following the negative gradient). The step size is controlled by the learning rate. Too large a step might overshoot the minimum, while too small a step can take a long time to converge.
Remember, the regplot function under the seaborn library and how it works? Wonder how the regression line is decided or visualized. The regplot function uses the same gradient descent mechanism at its backend. It depicts the line that minimizes the cost function, for linear regression, Mean Squared Error (MSE).



The red points indicate the actual points and the blue points are predicted points. The difference between two points indicated by dotted red lines are called the differences. These differences are indicated by Δ's . The mean squared error is derived as:

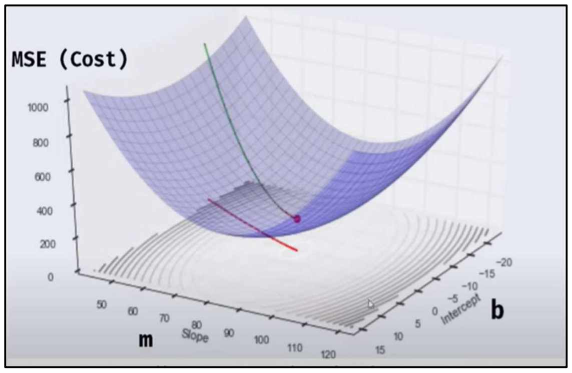
$$MSE = \frac{1}{n}\sum_{i=1}^{n} (\Delta_i)^2$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - y_{pred})^2$$

Commented [1]: The differences may be negative. To avoid negative value of MSE, squared term is used.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \square\, (y_i - \{mx_i + b\})^2$$

The gradient descent actually works in a three dimensional way. Its working can be visualized by the following graph:



In case of gradient descent, the steps are -

1. Firstly, the values of coefficients m and b are initially guessed. The guessed value is generally 0 for both of them.
2. The MSE is calculated using the zero initialized coefficient values.
3. Gradients are calculated. The same step is decreased in the coefficient values and then the cost function is calculated.
4. This process is continued unless the cost function is not minimizing further.

Mathematical Expression for Gradient Descent

Consider the Mean Squared Error (MSE) cost function, given as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \square\, (y_i - \{mx_i + b\})^2$$

Now, for calculating the slope of the cost function, we must determine the variables in it. There are two variables in the cost function, m and b respectively. So, the partial derivative of the MSE cost function w.r.t m and b will be:

$$\frac{\partial}{\partial m}(MSE) = \frac{2}{n}\sum_{i=1}^{n} \square - x_i(y_i - \{mx_i + b\})$$

$$\frac{\partial}{\partial b}(MSE) = \frac{2}{n}\sum_{i=1}^{n} \square - (y_i - \{mx_i + b\})$$

The partial derivatives give information about the slope, or the direction in which the point moves. Now, how much to move, is decided by updating the values of m and b. So, after updating m and b values will be:

$$m := m - \alpha . \frac{\partial}{\partial m}(MSE)$$

$$b := b - \alpha . \frac{\partial}{\partial b}(MSE)$$

Commented [2]: Here, alpha is called the learning rate

Python Sample Notebook for Gradient Descent

For this notebook, numpy and matplotlib libraries are used. A set of variables (x) is created as numpy array and corresponding y values are determined using the equation $y = 8x - 3$. The goal of this notebook is to apply a gradient descent mechanism on the data set. The initial values of m and b are guessed as 0. 10 data points are considered. Learning rate is 0.001 initially. The objective is to see the cost function decreasing with increase in iterations. At last the m and the b value will be very close to 8 and (-3) respectively. Adjust the learning rate and number of iterations to minimize the cost as wished. The same work mentioned above is accurately conducted in the Python notebook, attached with it.

Chapter - 4: Load and Save models using pickle and joblib

Once a model is created, now the second step is to save or store the model for further use. Thus, the model can be shared with some other clients in an efficient way. To complete this job, python offers two libraries. They are pickle library and the joblib module. Both are efficient ways to store that.

Pickle library stores the model using binary codes. In pickle, to write or save the model 'wb' is used with the intended model name to be shown. However, for using the model, specifically reading it 'rb' is used.

Joblib also provides the same service. However, in the model, if there are huge amounts of array, then it is a better way to use joblib, instead of pickle. The sample codes are given in a notebook. It contains the methods, codes and how to store and reuse the models.

Commented [3]: In previous time, joblib was a part of sklearn's external module. Few years back, joblib is pulled out from Sklearn. Now, it is completely separate module like pandas, NumPy.

Chapter - 5: Logistic Regression Model

Basics of Logistic Regression

In case of linear regression (simple or multiple), the target variable (output) must be a continuous variable. But what will happen if the output variable is categorical having two or more categories? In such cases, Logistic regression (Supervised Algorithm) models are used. If there are two categories in the output variable, then we opt for binary logistic regression, otherwise multiclass logistic regression is used.

Binary logistic regression is the most used method in the family of logistic regression. This process is utilized in various aspects such as detecting mail as spam or not, to determine whether a customer will buy insurance or not, which party a person is going to vote for etc. As the main goal of these models is to classify or assign a person or object in a particular class, this type of problem is commonly known as classification problem.

Types of Logistic Regression

In general, there are two types of logistic regression. They are -

- Binary Logistic Regression - the outcome variable has two categories - for example: a person will buy a car - yes/no; the selected person is male or female - yes/no
- Multiclass Logistic Regression - the outcome variable has more than two categories - for example: the object selected is of which color - red/green/black; which company car do you like - tata/suzuki/honda/bmw/audi/mercedes/mg/volkswagen/mahindra etc.

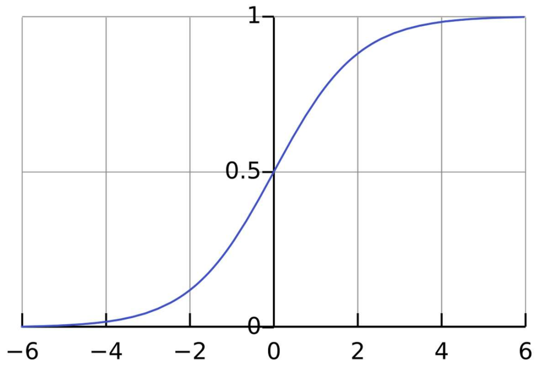
Mathematical Representation of Logistic Regression

In case of logistic regression, instead of the actual prediction, we predict the chance of that object/person to be in our desired class/category. So, logistic regression basically deals with the probability that an object or a person may belong to a certain class. As we know that probability values lie between 0 and 1, so the graph we get from logistic regression also depicts a s-like shape, called a sigmoid function. Mathematically this function is represented as:

$$sigmoid(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

In terms of graphical representation, a scatter plot of the sigmoid function is represented as:

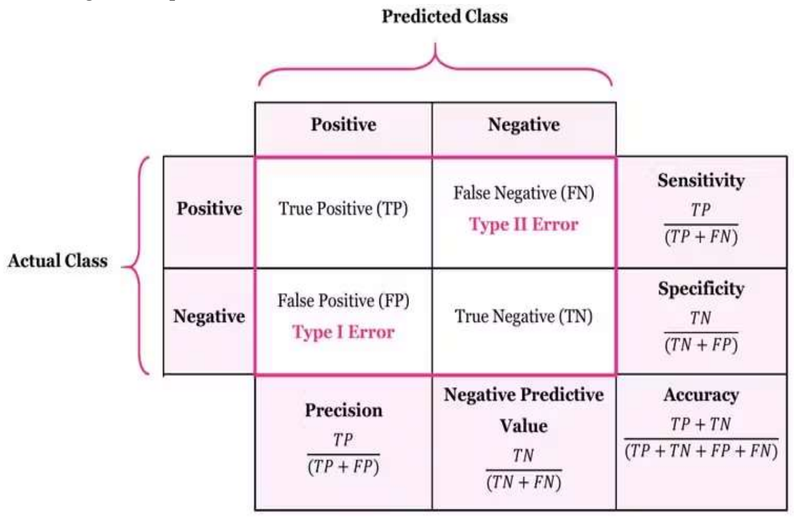
Commented [4]: The sigmoid function is often called as logit function



For a customer having the chance of buying a car, if the probability value is greater than or equal to 0.5, we claim that the customer is likely to buy a car, otherwise we claim the opposite.

Problem is Logistic Regression and Confusion Matrix

There are certain problems in logistic regression models, specifically when the number of data points increases. For example, after building a model it can be shown that some data points actually belong to class-A, but the model predicts that the points belong to class-B or vice-versa. In such cases this is called an error. Such situations can be dealt with using a concept called confusion matrix. The confusion matrix looks like this:



		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

True Positive (TP): When the actual and predicted outcome is same (positive - 1), no error
True Negative (TN): When the actual and predicted outcome is same (negative - 0), no error
False Negative (FN): When actual outcome is positive (1), but model predicts as negative (0), Type-II error
False Positive (FP): When actual is negative (0), but model predicts as positive (1), Type-I error
Evaluation metrics: The four evaluation metrics are also defined in the above picture. They are Precision, Accuracy, Specificity and Sensitivity. Sensitivity is also known as Recall. The harmonic mean of precision and recall is called F1 score.

Binary Logistic Regression Python Notebook

In this python notebook, a logistic regression model is built. The data set that is used for this model building is named ‘Weather_Data.csv’ that contains rain information of Australia over some time. The target variable is ‘RainTomorrow’, and there are lots of mixed types of input variables. However, as the process of dealing with categorical variables is already covered, the model is built by using only the numeric features. Sklearn’s modules such as linear_model, model_selection, and metrics are used for various purposes. The accuracy of model is close to 85%

Multiclass Logistic Regression Python Notebook

The data set that is used for this purpose comes as a built in data set in sklearn library. The name of the data set is ‘digits’. In this python notebook, a multiclass logistic regression model is built, where the main aim is to predict some handwritten digits using some images and their pixel values. This notebook contains the train test split part, evaluation metrics part and concept of confusion matrix.

Chapter - 6: Decision Tree

Basics of Decision Tree

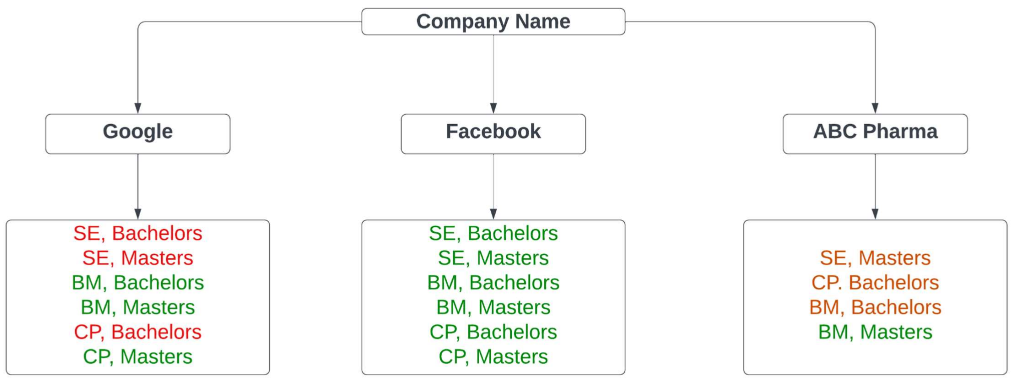
The main aim of using decision boundaries is the same as logistic regression. The only difference lies in the sparsity of the data. If the data can be separated efficiently by using a line of a s-like shape (sigmoid), then logistic regression is used. Now, often one may plot the scatter plot of some data points and observe that by using one such line/curve, separating data points is not possible. So, there are more than one decision boundaries. It is a Supervised Algorithm.

A decision boundary is a line/plane/curve/hyper plane-like structure that helps to separate two or more than two classes using a decision tree model. As the name suggests, it typically yields a tree-like structure.

Mechanism behind Decision Tree Algorithm

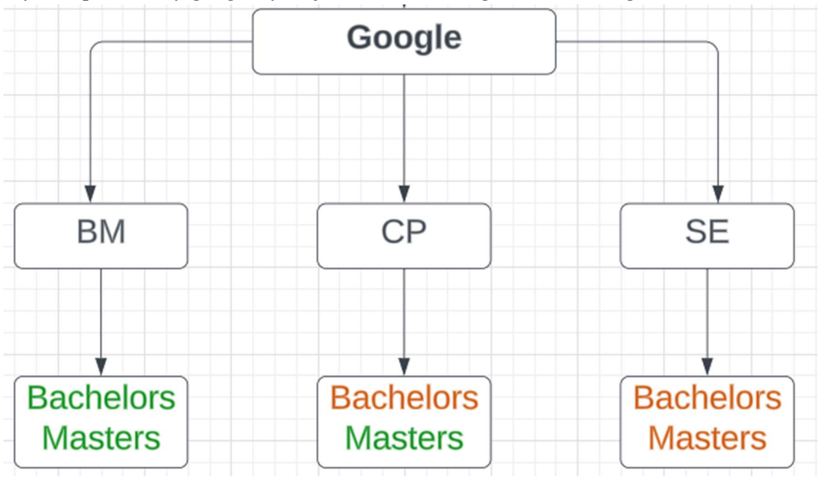
For this algorithm, the data set named ‘Employee_Salaries.csv’ is used. The data contains information about their company, job role, degree and their salaries. The ‘salary’ column is categorical. If the person’s salary is more than $100k, the value is 1, else the value is 0. The ‘salary’ column is the outcome/output/target variable for this decision tree model.

By observing the data, it is shown that there are three different companies. They are Google, ABC pharma and facebook. So, the model first separates the complete data by using the company name.

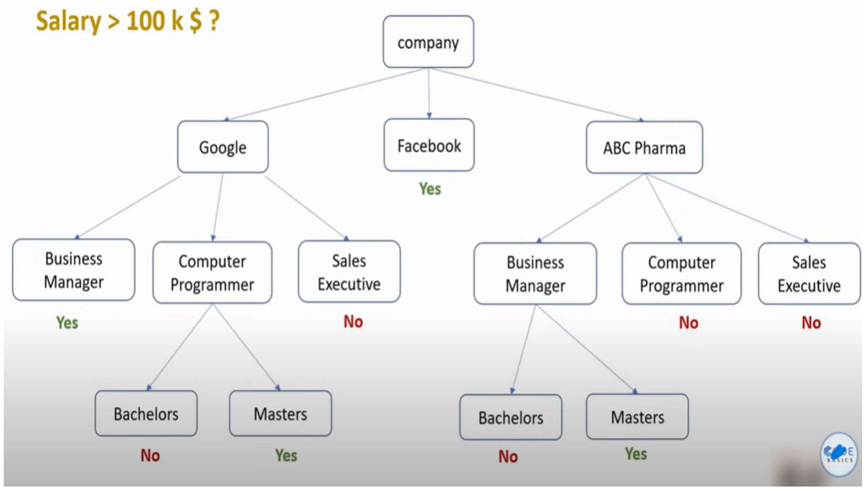


The green color means salary is more than $100k, red color means salary is less than $100k. So, it is evident that by separating the data company wise, we can easily get the final result for Facebook. However, for Google and ABC Pharma there are mixed types of salaries that need to be processed.

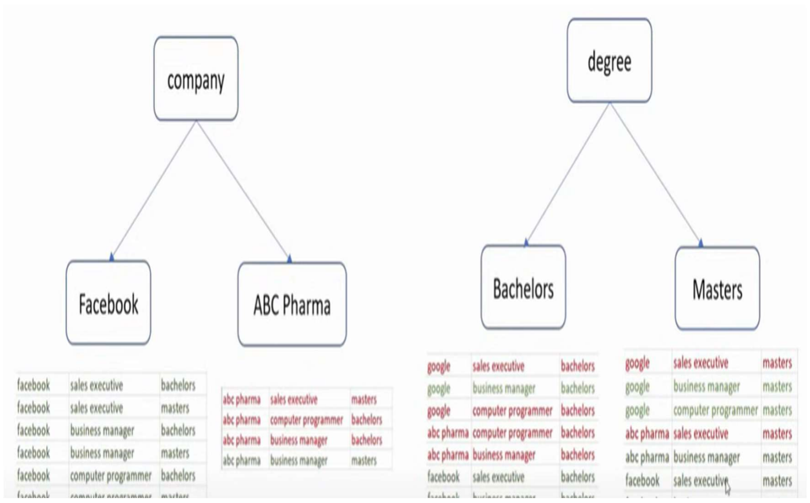
Again if we try to separate only google by its job roles, we will get the following results:



So, thus we can efficiently filter the Business Manager and Sales Executive role. However, the Computer Programmer role is still mixed. So, continuing the process for the complete data set, we can efficiently classify the data as below:



In real life cases, there may be 50 attributes. So manually splitting the data is impossible. Currently the splitting is initiated by company name. However, one may start the splitting by using job roles and end up separating the complete data. So, the order chosen for initiating the method affects the performance of the algorithm.
The term Entropy means measure of randomness. Low Entropy value suggests that the criteria of first separation is chosen correctly. This is evident from the below picture:



facebook	sales executive	bachelors
facebook	sales executive	masters
facebook	business manager	bachelors
facebook	business manager	masters
facebook	computer programmer	bachelors

abc pharma	sales executive	masters
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
abc pharma	business manager	masters

google	sales executive	bachelors
google	business manager	bachelors
google	computer programmer	bachelors
abc pharma	computer programmer	bachelors
abc pharma	business manager	bachelors
facebook	sales executive	bachelors

google	sales executive	masters
google	business manager	masters
google	computer programmer	masters
abc pharma	sales executive	masters
abc pharma	business manager	masters
facebook	sales executive	masters

As selecting company yields low entropy, so my information gain will be high

Decision Tree Model building Notebook

The 'Employee_Salaries.csv' data is used here. The python notebook shows how to build a decision tree model using python. Due to the lack of data points, train test split is not used. Instead the complete data set is used as training and test. The model result is predicted by using some imaginary entries. At the end, the tree-like structure is given to see how the separation is done.

Chapter - 7: Support Vector Machine (SVM)

Basics of SVM

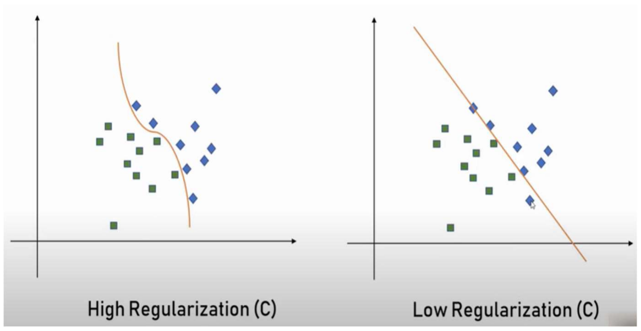
Support Vector Machine (SVM) (Supervised Algorithm) is a very popular classification algorithm. For a data set, it can be visually observed that the classes are separated. However, the problem is how to know which line best separates the classes? To resolve this issue, SVM comes in hand. For the chapter, the 'Iris_Flowers_Data.csv' file will be used.

One way to look at it is to calculate the margin (distance between the fitted line and original data points). Now the question is which line is a better choice? The one with low margin or the line with high margin? The line with higher margin is better as it best separates the data points. The separating plane for a 2D feature is just a straight line. For 3D features it is a plane. For nD features it gives a hyperplane.

Terms associated with SVM

Gamma: This is a concept in SVM that arises when the trend line is drawn by considering the data points. If the trend line is drawn by using the trend line near points, then we call this model a high gamma model. However, trend line margins can also be calculated by using the far points. In such cases, this is a low gamma approach. Using low gamma sometimes yields some problems with accuracy of the model.

Regularization: This is a concept used to make the model more generalized by making the model a good fit. Without this concept, there can be overfitting and underfitting. This is denoted by letter c.



Kernel: Suppose, we have X and Y direction in the graph. We are defining another variable called Z, such that Z can be denoted as: $Z = x^2 + y^2$. After this it can be seen that the Y axis vanishes. In actuality it is completely orthogonal to the plane of your screen, and so it can't be visible. The transformation Z is called a kernel.

Python Notebook for SVM

The iris data set is used for this SVM model building purpose. The python notebook contains detailed steps on how to build a SVM classifier along with how to check its accuracy. One thing to note thatML algorithms can't work with categories as input or output. So, where a categorical data set is present, use a label encoder or one hot encoder to convert it in a numerical form.

Chapter - 8: Random Forest Algorithm

Basics of Random Forest

Another supervised ML technology, popularly used for regression and classification tasks. The term 'Forest' in the model name actually depicts the tree-like formation of the decision tree. In the Random forest model, these trees are randomly created by the model and the best tree (model) is chosen from the huge number of possible decision trees.

How the Random Forest Model works

The original data contains mixed types of variables. Consider the 'Employee_Salaries.csv' file. The target variable is either 1 or 0. In the decision tree module, we observe that considering company name as first separation criteria, the model gives high information gain. For random forest, from the complete data, some batch data sets are created randomly. Then for each batch, a decision tree like model is built. As the selection of batch and decision tree building is a completely random process, hence the name is Random Forest Model. From each decision tree, we get a decision. Then from all the predicted decisions, the decision having majority vote is considered as actual result of the model

Random Forest Python Notebook

For this random forest model, sklearn's inbuilt data set 'Digits' is used. The python notebook contains all the necessary steps for building the Random Forest model, with train test split and evaluation. As we know, ML algorithms don't work with categorical variables, so label encoding may be useful where needed.

Chapter - 9: K-Fold Cross Validation

Basic Idea of Cross Validation

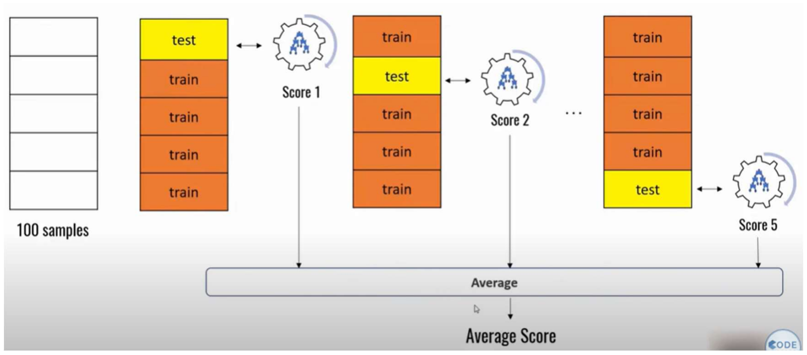
We can use ML models like logistic regression, decision tree, random forest, svm for classification tasks on the iris data set. However, which model is the best? In other words, out of these above four classification models, which one has the best performance? To resolve this question, a technique (not a ML model) is used known as cross validation.

Need for cross validation

The need of cross validation lies in the methods used for judging the accuracy of the model. There are many ways one can evaluate the performance of the model. They are -

- Build the model with the complete train and test data. Then judge the accuracy of the model with the same test data, used to build the model. The drawback of this method is that in this case, the model will show 100% accuracy, which is not liked in the ML field.
- The second method is to split the data set in train and test pat in a 80-20 or 70-30 ratio. The 80 or 70 part will be used to train the model. Then test the model with unseen 20 or 30 test data. This is a popular method. The drawback of this lies in discrepancies in train and test parts. Suppose, the train part contains information for a particular class, and the test part contains information of another class. Then the ML model fails the accuracy test as it is not trained on the test class.
- The third method is called K-fold cross validation. Suppose, there are 100 samples. So, these 100 samples are divided into 5 folds (5 fold cross validation). In the first iteration, the first fold is used for testing and the last four folds are used for training and the model score is noted. This process is continued until the last fold is used as a test sample. The average model score is calculated by averaging the model scores of 5 iterations.

Pictorial description of this method is given as:



K-fold Cross Validation Notebook

Sklearn's inbuilt data set, 'Digits' is used for this purpose. Firstly, a logistic regression, a svm and a random forest model is built completely with the data set. The code is given in a notebook. Cross validation model is built and the above three models are compared.

Chapter - 10: Naive Bayes Classification Model

Basic idea of Probability

Naive Bayes is a supervised ML technique that deals with probabilistic modeling. We all know that tossing a coin gives either a head or a tail with an equal probability of 0.5. In a similar way, getting a 3 when a dice is rolled is 0.167. This is the basic idea of probability. If, getting a 3 while throwing a dice is an event, say A, then mathematically we can say that $P(A) = 0.167$

Basic idea of Conditional Probability

The more advanced step of this probability concept is Conditional probability. It is the probability that depends on a previous event. For example, suppose picking a queen from a set of cards is an event Q, then $P(Q) = 0.0769$. This happens as there are four queens and a total of 52 cards. Now if i ask that i pick a random card from a set and found it to be a card of diamond, then what is the probability of getting a queen? The answer will be 0.0769. As there are 13 cards of diamond and there is only one queen. This probability is called conditional probability.

Suppose, A and B are two events. A happens at first. After A B happens. Then the probability of occurring B depende on A. Mathematically this is given as: $P(B|A)$ spelled as probability of B, given A. Similarly $P(A|B)$ is the probability of A given B.

Bayes Conditional Probability

Thomas Bayes gave a very popular expression regarding the conditional probability of two events. Suppose, A and B are two events. Then the conditional probability of A, given B has already occurred is given as:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

The term, $P(A|B)$ is called as posterior probability, the term, $P(B|A)$ is called the likelihood, the term $P(A)$ is called the prior probability, the term $P(B)$ is called the Marginalization

So, for the diamond queen problem, the task is to find $P(Queen|Diamond)$. According to the Bayes theorem, this is same as:

$$P(Queen|Diamond) = \frac{P(Diamond|Queen) \; * \; P(Queen)}{P(Diamond)} = \frac{\frac{1}{4} * \frac{1}{13}}{\frac{1}{4}} = \frac{1}{13}$$

Some Use cases of Naive Bayes Model

The Naive Bayes method is pretty much used in email spam detection, handwritten digit detection, weather prediction, face detection and news article categorization

Naive Bayes Python Notebook for Titanic Data set

The data set that is used here is called the 'Titanic_Data.csv'. The main purpose of this python notebook is to determine the passenger survival rate by using some demographic information and some other ship related information. So, mathematically this is same as:

$$P(\frac{Survival}{Gender, Class, Age, Cabin, Fare})$$

The method is called Naive Bayes as we assume the Naive assumption that the features such as Gender, Class, Age, Cabin and Fare are independent of one another. For this notebook, Gaussian Naive Bayes is used. This is used when the data distribution is normal. The train test split is done, the model is defined, trained and tested. The accuracy is calculated using the sklearn's metrics module.

Naive Bayes Python Notebook for Email Spam Detection

For this sample Naive Bayes model 'Email_Spam_Data.csv' file is used. This Python notebook contains the basic ideas of reading data, model building, train test split, and evaluating the model. The extra topics that are covered in this topic is count vectorizer, Multinomial Naive Bayes model, pipeline.

	Aniket	Chakraborty	I	am	my	is	surname	last	name
Doc-1	1	0	1	1	0	0	0	0	0
Doc-2	1	1	1	1	0	0	0	0	0

Sklearn's

Doc-3	0	1	0	0	1	1	1	0	0
Doc-4	0	0	0	0	1	1	1	1	1

CountVectorize: Count vectorize is a method to convert text, sentences into number form, by converting them into matrix format. The count vectorize method looks for similar words in sentences to build a matrix. For example, consider the following sentences: 'I am Aniket', 'Aniket Chakraborty I am', 'Chakraborty is my surname', 'Surname is my last name'. The distinct words from these four sentences are: 'Aniket', 'Chakraborty', 'I', 'am', 'my', 'is', 'surname', 'last', 'name'. So, this whole information can be converted into a matrix like structure, just like the above table.

Chapter - 11: K-Nearest Neighbor Algorithm

Basics of KNN Model

KNN or popularly called as K-Nearest Neighbors is a supervised ML algorithm, known for its simplicity in the field of classification problem. For building this model, 'Iris_Flower_Data.csv' file is used. After building a classification model, KNN helps to predict the class of a new object by measuring the Euclidean distance of the new data point from its neighboring data points. The k value in KNN indicates the number of data points to be taken as reference from the new data point. There is no fixed method or algorithm to determine the k value. It is determined by using trial and error methods. However, most models use k value as 5, but it completely depends on the user.

Now suppose, the value of k is 10. Then we take 10 nearest data points of the new data point introduced. Assume that out of these 10 data points, 7 corresponds to class-A and 3 corresponds to class-B. Then, the model predicts that the newly introduced data point will be of class-A.

Choosing the value of k is pretty important in the KNN algorithm. If the k value is very small or very high, there is a chance that the model starts misclassifying the data points. As a result, the accuracy of the model will go down.

KNN Python Notebook

The data set that is used is iris data. The notebook contains data exploration, train test splitting, model building and checking the accuracy of the model. The preferable k value is 3 for this notebook. However, for k values in the range 1 to 20, how the model behaves is observed by plotting their respective accuracy score for each k values.

Chapter - 12: K-Means Clustering Model

Basic Idea of K-Means Clustering

Clustering, specifically K-Means clustering, is an unsupervised ML algorithm that deals with forming clusters from data points. It is similar to group formation in any object. As it is an unsupervised algorithm, this type of ML algorithm doesn't need train test set splitting. The data used for unsupervised learning doesn't explicitly mention the target or input features. Even if the class labels are not provided. Only a set of features are provided. The main aim of this clustering algorithm is to form clusters from those feature data points. In this type of unsupervised learning, we generally go for obtaining underlying features in the data set.

Mechanism of K-Means Algorithm

The k in K-Means algorithm is a free parameter. Before building the model, the value of k must be supplied to the model for best performance. The data points are clustered into segments by observing the centroids of the clusters. If a point is near a cluster centroid, then the point belongs to that cluster. The main problem arises while determining the value of k. However, there is a method called the Elbow method. By this method, we can determine the perfect value of k.

Working of the Elbow Method

Suppose, we built the model with k = 2, therefore there are two clusters. Then we calculate the Sum of Square Error (SSE) for each cluster. For a cluster say cluster-1, the SSE is calculated as:

$$SSE_1 = \sum_{i=1}^{n} dist(x_i - c_1)^2$$

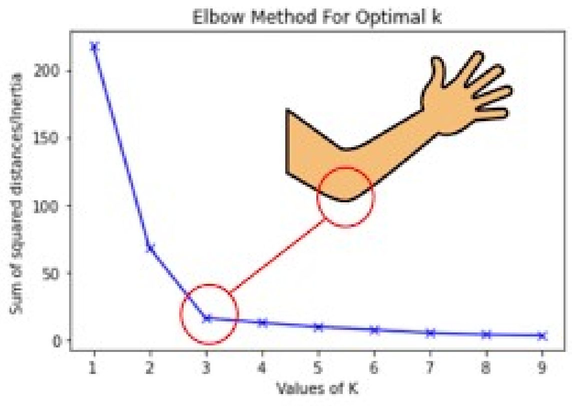
In the above expression, x_i are the i data points and c_1 is the centroid for cluster-1. In a similar fashion, for cluster-2 SSE_2 is calculated as:

$$SSE_2 = \sum_{i=1}^{m} dist(x_i - c_2)^2$$

After calculating Sum of Squared Error (SSE) for each cluster, they are summed to get Total Sum of Squared Error (TSSE) as:

$$TSSE = SSE_1 + SSE_2 + \ldots + SSE_K$$

Once we have the SSE values for k clusters, we plot the SSE values w.r.t to the k values. This eventually gives an elbow-like structure, hence commonly known as the Elbow plot. From the Elbow plot, the k value from which the SSE score takes a big slope, we say that k value is optimum for the K-Means model. For example consider the following picture:



Line plot between K and inertia

For k value 3, the model gives best performance.

K-Means Python Notebook

The data set that is used in this model building is '[Income_Cluster_Data.csv]'. This python [notebook] contains all facts, codes and techniques to build a K-Means clustering algorithm. Sklearn is used with proper Elbow plot and accuracy checking. No train test splitting is needed. No Separation of input and output variables are needed for this unsupervised model.

Chapter - 13: Principal Component Analysis

Basic Idea of PCA

Principal Component Analysis, popularly known as PCA is an unsupervised ML technique, used to reduce the dimensionality of a data. In the case of ML algorithms , we are often confused which features to take as input. Increasing the number of unimportant features will decrease the model performance. So, selecting important features is essential. To make this easy, PCA is used in almost all ML algorithms. PCA has two major roles in ML algorithms. They are:

- Easy model building and inferences
- Data visualization has become easier.

PCA creates new features called PC1, PC2,... by using linear combinations of previous features. For 100 features, one can create 100 principal components in descending order of their importance on the output variable. While applying PCA, the features must be scaled. The accuracy of the model may be compromised in this complete process.

PCA Python Notebook

In this exercise, sklearn's inbuilt data set digits data is used. In this Python notebook using the digits data, a logistic regression model is built having accuracy of 96%. Then PCA is applied to minimize the feature number to 29 from 64. However, the model performance reduces only 3%. This depicts that almost the complete variability in data is captured by those 29 features. However, if we build a model with only 2 PCs, the model accuracy decreases to 60%, a 33 % decrease in model performance.

Chapter - 14: Hyper-Parameter Tuning in ML

What are Hyper-parameters

Hyper parameters are those parameters of a ML model that can be tweaked or tuned by the user as per the requirement of the model outcome. For example, in gradient descent algorithm, the learning rate, denoted by the symbol α is a hyper parameter.

Need of Hyper-parameter Tuning

Suppose, we are dealing with the Iris flower data set and the aim is to build a classification model. Now, the question is which model to use for this classification problem? Logistic regression, Decision Tree, Random Forest, SVM, Naive Bayes are all possible choices. It is nearly impossible to build each and every model and then compare their accuracy. The process of choosing the optimum parameters for a ML problem is called Hyper-parameter tuning.

Sample Code for Hyper-parameter Tuning

The Iris flowers data set is used here. Firstly a SVM model is built that yields an accuracy of 96%. The SVM model has many hyper parameters such as 'C', 'kernel' etc. One way is to make a for loop for all these variables and then make the model. But as the model hyper parameter increases, the for loop becomes more dense and as a result, the code becomes complex. So, to resolve this issue, GridSearchCV is used, where by writing one line of code this whole process can be summed up. All these methods are shown in this notebook. However, there are some limitations of the GridSearchCV method. It is the computation cost. For example, instead of supplying 'C' value 1, 10, 20, if the 'C' value lies in the range 1 and 50, then the computation cost increases. At last, the dir() method can be used to get the model hyper-parameters of a specific model. The GridSearchCV method tries all permutations and combinations of the supplied hyper-parameters and then gives the result after seeing all combinations. However, in sklearn, there is another class called RandomizedSearchCV, that doesn't go through all permutations and combinations. Thus it makes the computation cost less. Also, we came to the conclusion that SVM is the best model for Iris flower data classification.

Chapter - 15: Regularization in ML

What is Overfitting?

Overfitting in machine learning occurs when a model becomes too complex and learns not only the underlying patterns in the training data but also the noise or random fluctuations. As a result, the model performs very well on the training data but fails to generalize to new, unseen data, leading to poor performance on the test or validation sets.

Key indicators of overfitting

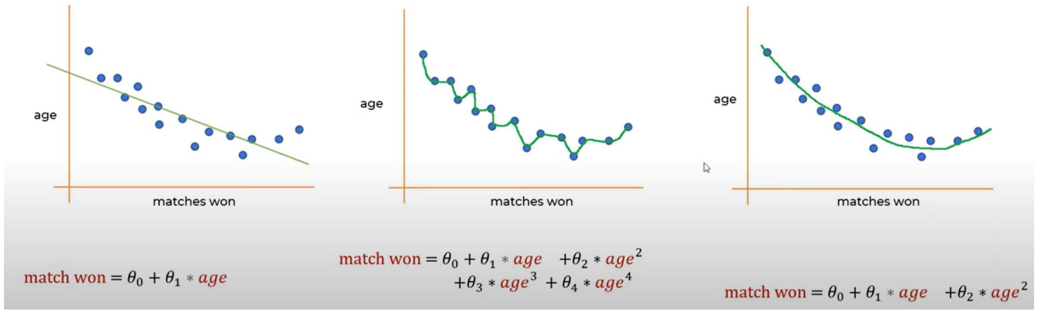
1. **Low training error and high test error:** The model is highly accurate on the data it was trained on but performs poorly on new data.
2. **Excessive complexity:** Overfitting often happens when the model has too many parameters or when it's too flexible relative to the amount of training data available.

Ways to resolve overfitting

- Regularization (e.g., L1, L2 penalties)
- Cross-validation
- Simplifying the model by reducing the number of features or parameters
- Early stopping during training, Increasing the amount of training data

Understanding Overfitting using graph

Suppose, we are trying to predict the number of matches won by an athlete with the increase of his age. In general, as age increases, due to fatigue, the number of matches won decreases. Consider the following graphs to understand overfitting.



Graph-1: It is a linear regression trend line graph. It is very simple. The line best estimates the data points shown in the graph. But the problem is, if a data point is far from the line, the meaning of this graph becomes vague as the line will not estimate the far data point. This is an example of poor fitting, known as under fitting.

Graph-2: The graph estimates the data points with high accuracy and high precision. It is a zig-zag line that goes through all points. The equation given is very complex. The problem arises when new points are added or seen. The graph can't estimate the new point due to its complex nature. This is an example of overfitting.

Graph-3: Graph doesn't connect all data points, but efficiently dreams a line which can estimate all present data points and all unseen data points. This line perfectly fits all the data. This is known as perfect fit.

Solving Overfitting using Regularization

Consider the overfitting model with the overfitted equation given as:

$$matches\ won \ = \ \theta_0 + \theta_1 * age + \theta_2 * age^2 + \theta_3 * age^3 + \theta_4 * age^4$$

The idea behind resolving overfitting is to somehow make the coefficients θ_3 and θ_4 zero, so that the equation reduces to the form:

$$matches\ won \ = \ \theta_0 + \theta_1 * age + \theta_2 * age^2$$

In general, it is not possible to make θ_3 and θ_4 exactly zero, but we can shrink these coefficients very close to zero, so that we can ignore them while building the model.

Types of Regularization

Regularization can be of two types. They are:

1. Lasso Regularization or L1 Regularization
2. Ridge Regularization or L2 Regularization

Lasso and Ridge Regularization are techniques used to prevent overfitting by adding a penalty to the regression model's cost function. Both are forms of regularization, which helps in shrinking the model coefficients, but they have different approaches in how they apply the penalty.

Lasso or L1 Regularization

Mathematical Expression: In case of linear regression model, the cost function is given by the metric MSE, called as Mean Squared Error and it is given as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - y_{pred})^2$$

To penalize this cost function a regularization term is added. This is given as:

$$Cost\ Function_{Lasso} = MSE + \lambda \sum_{i=1}^{n} |\beta_i|$$

In this expression, λ is called the regularization strength and β_i's are the coefficients of the linear regression model.

Use Case of L1 Regularization: Lasso tends to shrink some coefficients to exactly zero, effectively performing feature selection by excluding less important features from the model. If you suspect that many features are irrelevant or redundant, Lasso will help in automatically selecting the most important features by setting less important ones to zero. It is useful when you want a sparse model. So, L1 regularization is useful, when we already know, which features are not important from previous experience for model building.

Ridge or L2 Regularization

Mathematical Expression: In case of linear regression model, the coast function is given by the metric MSE, called as Mean Squared Error and it is given as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - y_{pred})^2$$

To penalize this cost function a regularization term is added. This is given as:

$$Cost\ Function_{Lasso} = MSE + \lambda \sum_{i=1}^{n} {\beta_i}^2$$

In this expression, λ is called the regularization strength and β_i's are the coefficients of the linear regression model.

Use Case of L2 Regularization: Ridge shrinks the coefficients closer to zero but does not set them to zero. It generally retains all the features, just shrinking their impact. When you believe that all features contribute to the output, but you want to prevent overfitting by shrinking the coefficients. It is useful when there is multicollinearity (correlated features), as Ridge will distribute the impact of the correlated features. So, L2 or Ridge regularization is used when one doesn't know which features are not important. So, uniformly shrinking all coefficients to zero is a better approach.

Regularization Python Notebook

For regularization notebook, a data set named '[Melbourne_HousePrice_Data.csv]' file is used. This [notebook] contains basic data exploration, preprocessing, manipulation steps with a linear regression model building for 'price' column as target. The train test split is done by considering test size 30% and selecting the state as 2. It is shown that the model performs approximately 68% on the train data, but performs only 13% on the test data, which clearly indicates overfitting. Lasso (L1) and Ridge (L2) regularization are performed and it is seen that both regularizations increase the test set performance to approximately 68%.

Chapter - 16: Bias and Variance in ML problem

Concept of Variance

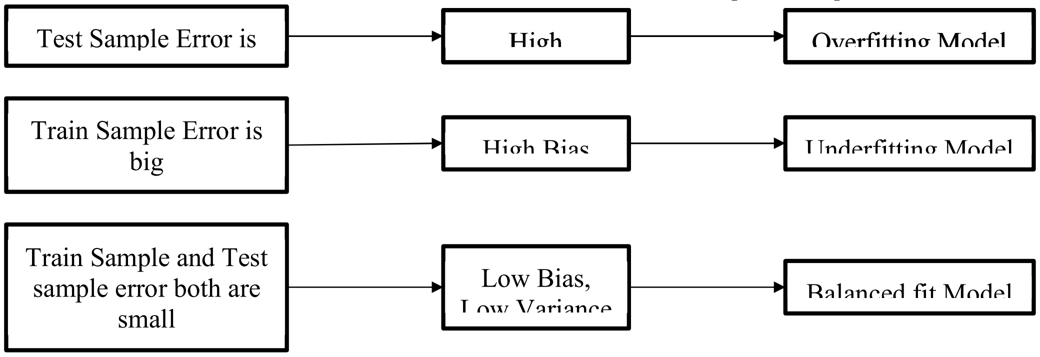
As discussed above, overfitting, underfitting or balanced fitting completely depends on the train test split method. As the train test split method splits the sample randomly, so it is a possibility that for the first run, one may get a test error of say 100 (very much overfitting). On the other hand, another data scientist got a test error of say 27 (low overfitting). Both of these results depict the same intuition of overfitting. This is often called high variance. The test error varies randomly as samples are chosen randomly. So, overfit models can be distinguished by the fact that they have high variance. In a similar fashion, if the test error doesn't vary too much even after the random sampling of train and test samples, it is called as low variance.

Concept of Bias

The concept of bias relates to the train error. If the train error varies by choosing random samples, then we can say that this is a problem of high bias. Bias is a measurement of how accurately a model can capture a pattern in a train set.

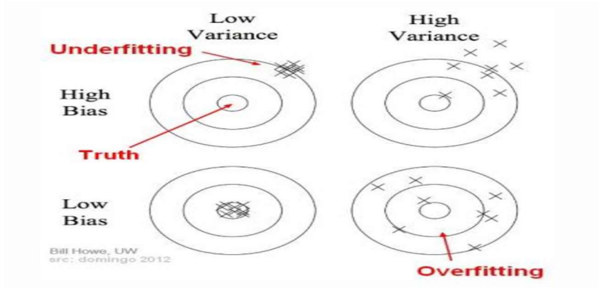
Example

In the previous overfit problem of 'Melbourne_HousePrice_Data.csv', the bias was low as train error was close to zero. So, higher the train error, higher the bias. If for a model, the train error and test error are low, then the model is called a low variance, low bias model. So, these information can be depicted as a picture:



Bias and Variance using Bulls-Eye Diagram

By using a Bulls-eye diagram, we can easily spot the overfitting and underfitting in a model as:



The center part of each circle is called the truth values. If the data points are near the center, it is called low bias, otherwise if the data points are far from the center, it is called high bias.
If the data points are grouped together like a cluster, it is called low variance. If the data points are scattered, it is called a high variance.

Chapter - 17: Ensemble Technique in ML

Basics of Ensemble Technique

Ensemble is a technique that is used when there is a high variance problem. In this technique, we use multiple models, we train them with training samples, we test them with test samples and compare the results. After this we combine the results from different models and get our final model. The main concepts that are used in Ensemble technique are Bagging and Boosting respectively.

Concept of Bagging

Suppose, there are 100 samples and use those samples as a ML model train and test set. The problem is the model depicts high variance - overfitting. So, to resolve this issue, we decrease the sample size to 70 by random resampling with replacement technique. So, in this technique, we may get the same data point multiple times. Using the same approach, n number of such small data sets are created from the original data set, having 70 data points each. The model is built on these n small subsets and they are compared. After prediction, we get the result from n small sample sets and we take the majority vote.

The main advantage of this method lies in their model nature. The models that are built by using the n small sample sets are weak learners as the number of data points decreases. So, it is more likely that those n models will not overfit. This technique can be used for linear regression or logistic regression problems. This technique is also known as Bootstrap aggregation as the procedure of creating a small set of samples using resampling with replacement is called Bootstrap.

Random Forest algorithm is one technique under the ensemble method as discussed earlier. It has some distinctive facts also. In the case of the Random Forest algorithm, we not only make small samples using the rows, we also make small samples using the features.

In Bagging technique, the underlying model can be SVM, KNN or logistic regression, Whereas, Random Forest is an example of Bagged tree. In general, Bagged tree means where each model is a tree.

Ensemble Python Notebook

For this ensemble technique method, the 'Pima_Indian_Diabetes_Data.csv' file is used. This python notebook deals with basic data exploration and making a decision tree model with 5 fold cross validation. The accuracy of the model is 71%. Then Bagging Classifier is used with a decision tree estimator, 100 samples made with 80% of the data. The obb_score is obtained from the model. This comes to 75%. Then, using the same cross validation method, a RandomForest classifier is run and the model shows 77% accuracy.

Chapter - 18: Feature Engineering in ML

Basics of Feature Engineering

Feature engineering is the technique used in ML that helps to create one or more essential features from existing features. For example, If 'Height' and 'Weight' of a person is given in a data and they are impactful on some outcome, then another feature 'BMI' can be created by calculating $Weight/(Hight)^2$. It is a possibility that 'BMI' is also impactful in this case. So, if 'BMI' is impactful, then we can easily drop the 'Height' and 'Weight' column. This is the main essence behind Feature Engineering.

However, feature engineering is not only the process of creating new columns. It includes detection and handling of outliers, missing value detection and handling, OneHotEncoding some categorical columns etc.

So, in one line, Feature engineering is the process of extracting useful features from existing features of a data set by using Mathematics, Statistics, Economics, Business knowledge, domain specific knowledge.

Chapter - 19: Outlier detection and handling in ML

What are outliers?

Outliers are some unusual data points in a data set that are very different from the rest of the existing data points. Suppose, we are dealing with a human age data set that limits the age up to 95 years. If there is a data point in the data set that has age 1000, then the data point is very different from the rest of the observations and is called an outlier.

Concept of Percentile

If a person's Mathematics score of 56 has a percentile value of 50%, then we can say that 50% of the samples are below 56. So, 50% of total students score less than 56 in Mathematics.

Percentile Python Notebook

For this notebook, the 'Person_Heights_Data.csv' and 'Bangalore_PropertPrice_Data.csv' files are used. This python notebook contains basic data exploration, basic statistical summary check, use of percentile to detect outliers, get the outlier free data set etc.

What is Z-Score

Z-score is a standardization technique used in ML to detect and handle outliers. The concept of Z-score is actually opposite of the Standard deviation. For example, for a random sample X (say), if a data point is 3 standard deviations away from its mean, then the Z-score of that point will be 3. Z-score is typically denoted by the symbol Z and it is calculated by using the standard deviation σ and the mean of the sample μ and the particular data point say x. The Z-score is calculated as:

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

What is Standard Deviation

Standard deviation, also abbreviated as std is the measurement that depicts how far away the data points are scattered from their mean value. Standard deviation is the positive square root of deviance. For a sample X (say), if the mean is denoted by $\underline{x}$ and the data points are denoted as x_i for $i = 1,2,3,\ldots,n$, then variance, denoted by σ^2 is defined as:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} ⬚ (x_i - \underline{x})^2$$

So, standard deviation is defined as:

$$std = \sqrt{\sigma^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} ⬚ (x_i - \underline{x})^2}$$

Z-Score and Standard Deviation Python Notebook

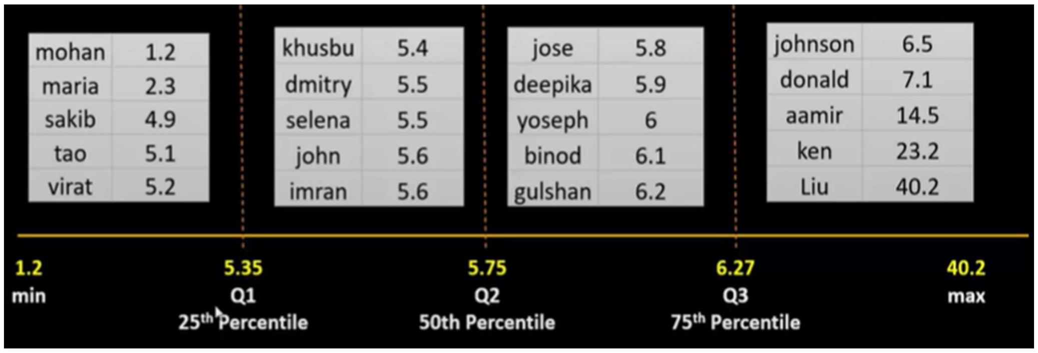
For this sample work, the 'Weight_Height_Data.csv' file is used. This notebook deals with how to detect and handle outliers from a data set using Z score and standard deviation method. This notebook also gives a comprehensive idea on normal distribution and bell shaped curve with how to plot a histogram along with the bell shaped curve as density line. It is observed that in industry level, to detect outliers for large datasets, a method called 3 standard deviation (Z score is ± 3) is used.

Interquartile Range (IQR) and Outlier Detection

The idea of IQR, comes from the idea of Percentile. The types of percentiles are:

- Q1(first percentile:25%): 25% of samples are less than that number
- Q2 (Second percentile:50%): 50% samples are less than that number
- Q3 (Third percentile:75%): 75% samples are less than that number

Using an image this concept can be clarified in a simple way:



The IQR, is defined by using the values of Q1 and Q3 as:

$$IQR = Q3 - Q1$$

Using this IQR values, a lower limit and upper limit is defined as follow:

$$lower\ limit = Q1 - 1.5 * IQR$$

$$upper\ limit = Q3 + 1.5 * IQR$$

IQR Python Notebook

For this sample code, the 'Person_Heights_Data.csv' file is used. The notebook contains detailed methods on how the quartiles are determined, IQR is calculated, how to calculate upper limit and lower limit using IQR and then using those upper and lower limit values, the outliers are detected and cleaned from the data set.

Chapter - 20: Python ML Project - Real Estate Price Prediction

Introduction

In this project, we are going to use the 'Bangalore_HousePrice_Data.csv' file for creating the model. Different cool data science stuffs like data cleaning, feature engineering, handling missing values, outlier detection & handling and finally building the linear regression model. Use GridSearchCV for better model performance. Save the complete model using pickle and then create a Python flask server to interact with it.

Going through the data set

The data set has 8 columns, out of which the 'Price' column is the target variable. Rest of the variables are input variables. The 'Price' column is in Lakhs. So, the value of 39 in the 'Price' column means 39 lakhs Indian rupees. As the input and output variables are clearly stated, it will be a supervised learning problem.

Python Project Steps

The Python ML project is done with google colab. The project steps are listed below: The python notebook for this project is hyperlinked.

1. **Connecting google colab with google drive**
2. **Importing all required packages**
3. **Basic data exploration steps:**
    a. Reading the data using python
    b. Check the first five rows
    c. Get the shape of the data frame (Rows and Columns)
    d. Get the column names of the data frame
    e. Get the count of Houses, grouped by area_type
    f. Get the column types of the data frame
    g. Drop useless columns 'area_type', 'balcony', 'availability' and 'society'
4. **Cleaning steps for data frame:**
    a. Check for missing values (NA) values in the data frame

   b. Drop the rows that contains the missing values
   c. Removing discrepancies from the 'size' column by defining function and converting it into an integer object as a new column 'BHK'
   d. Handling discrepancies in the total_sqft column, converting range to numbers and drop the rest

5. **Feature Engineering steps:**
   a. Make a deep copy of blr_data4 in to blr_data5
   b. Create a new column called 'price_per_sqft' as price/total_sqft
   c. Explore the length of unique entries of the location column
   d. Write a function to strip any extra spaces from the entries of location column
   e. Get a summary statistics of group by locations in descending order to get the number of data points per location
   f. Change the location name to 'other' for the locations having less than or equal to 10 houses
6. **Outlier detection and Outlier handling steps:**
   a. Check the shape of blr_data5 data frame
   b. Make a deep copy of blr_data5 as blr_data6
   c. Calculate the toatl_sqft_per_bedroom column in blr_data6
   d. If the column value is less than 300 (confirmed by my manager), then it is an outlier
   e. Remove it from the data frame
   f. Removing houses that costs more for 1 BHK than 2 BHK, same for 2 and 3 BHK
   g. Remove houses that has number of bathrooms +2 than bhk
7. **Preparing the data for model building:**
   a. Make a deep copy of blr_data9 as df
   b. Check the columns of the data
   c. Drop unnecessary columns
   d. One hot encode the location column, delete the location column, drop the other column in dimmies
8. **Building the model:**
   a. Separate input(X) and output(y): output is price
   b. Train test split the X and y using 20% test size and random state 10
   c. Build a linear regression model as lr_model: define, tain, test
   d. Use Cross validation and GridSearchCV to find best model
   e. Save the model as 'RealEstate_Model' using pickle
9. **Building a Python Flask Server in PyCharm:** Create a file called 'Code' in C drive. Inside C drive, create three folders namely 'Client' for other user connections, 'Server' for the Python Flask server and the 'Model' where we store the ipynb file along with .pickle and .json files.

Chapter - 21: Python ML project - Image Classification Problem

Introduction

In this project, we are going to use the image files for creating the model. Different cool data science stuffs like data cleaning, feature engineering, handling missing values, outlier detection & handling and finally building the classification model. Classification is also a supervised ML technology. In this project methods for scraping images from the web are covered with the use of OpenCV library (popular for computer vision level image

Commented [5]: It is a better choice to use Jupyter notebook for this image classification task processing). After completing the model, the model is saved as a pickle file. A python flask server is created and then html, css and javascript are used to build a website. The Python notebook is hyperlinked here for use.

Python Project Steps

The steps for this python project are:

1. **Collection of Data:** In the context of data collection for image classification tasks, there are four ways of doing that. They are:
    a. Manually downloading images from google. This process is very labor intensive and needs time to complete.
    b. Use Python for web scraping. An article is provided for this process. But the main problem is there is no certainty that it will work in future. The reason being the update of google security features. Web scraping is a very gray area. So, even if the information is public, google doesn't want some bots to web scrape their content.
    c. The next and probably most secure way is to use a chrome extension called Fatkun. It provides easy search and download images and allows customization as required.
    d. The last way is to buy those images from a third party vendor.

    In this project, five sports celebrity images are used: Virat Kohli, Lionel Messi, Serena Williams, Maria Sarapova and Roger Federar. The respective images are given in the GitHub repository.
2. **Data Cleaning steps:** Any data scientist requires more than 80% of their time and effort in the data cleaning process. The images that are downloaded from the web have a lot of issues or discrepancies inside them. Some of them may be hazy, some of them may be blurred etc. So, if in a face, two eyes are properly seen, then keep that image. Otherwise don't consider them or process them using OpenCV. The Haar Cascade method is used for this purpose. It crops all the images from the provided set of images having two eyes. After that we run a manual data cleaning process, in which we delete unwanted images. The notebook, hyperlinked here contains all necessary details of the steps described above.
3. **Feature Engineering Steps:** Feature engineering is the most interesting step in a data science project. It deals with extracting the useful features from a completed set of features that will surely impact the result. For this project, a technique called Wavelength transform is used as feature engineering. In the wavelength transform method, the complete image is converted into a black and white image, in which different areas and places can be differentiated very easily. The notebook is hyperlinked here.
4. **Model Building:** In this python notebook, the SVM model with 'rbf' kernel is used for building the image classification model. Sklearn's pipeline is used to build a customized model. Scaling is done to remove all discrepancies. For fine tuning the model, GridSearchCV is used. The model shows 88% accuracy on test samples without using neural networks. A classification report is calculated. Confusion matrix is plotted for the best model.