جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الانظمة الطبية الذكية

المرحلة الثالثة

Subject: Artificial Intelligence AII

Class: Third

Lecturers: Dr. Muneera Abed Hadi, M. Sc. Rouaa Safi , M. Sc. Ansam Ali

University of Information Technology
and Communications
College of Medical Informatics
Intelligent Medical Systems Department

Machine Learning Course

Lecture Four

By: Dr. Muneera Abed Hmdi

Key Steps for *Training a Supervised Learning Model*

- ***Below are the key steps involved in training a model for supervised machine learning:***
- ***Data Collection and Preprocessing***: Gather a labeled dataset consisting of input features and target output labels. Clean the data, handle missing values, and scale features as needed to ensure high quality for supervised learning algorithms.
- ***Splitting the Data***: Divide the data into training and test using cross validation technique such as holdout cross validation with training set (80%) and the test set (20%).
- ***Choosing the Model***: Select appropriate algorithms based on the problem type (C/R). This step is crucial for effective supervised learning in ML.
- ***Training the Model***: Feed the model input data and output labels, allowing it to learn patterns and relationship by adjusting internal parameters.

Feature Scaling Methods

- ***Normalization and standardization*** both belong to the idea or category of feature scaling. Feature scaling is an important step in preparing data for machine learning models.

- It involves ***transforming the values of features in a dataset to a similar scale***, ensuring that all features contribute equally to the model's learning process.

- ***Feature scaling*** is important because, when features are on vastly different scales, like a feature ranging from 1 to 10 and another from 1,000 to 10,000, models can prioritize the larger values, leading to bias in predictions. ***This can result in poor model performance and slower convergence during training.*** Feature scaling addresses these issues by adjusting the range of the data without distorting differences in the values.

Normalization

- ***Normalization*** rescales feature values within a predefined range, often between 0 and 1, which is particularly useful for models where the scale of features varies greatly.

- Most widely used normalization method is ***z-normalization*** (standard score). If ***mean μ and standard deviation σ of the data is known***, z-normalization is calculated as $z = x - \mu \sigma$.

Standardization

- ***Standardization*** centers data around the mean (0) and scales it according to the standard deviation (1). This process adjusts the feature values by subtracting the mean and dividing by the standard deviation.

$$X_{std} = \frac{X - \mu}{\sigma}$$

- Where:
  - X is the original value,
  - mu is the mean of the feature, and
  - sigma is the standard deviation of the feature.

- ***This formula rescales the data in such a way that its distribution has a mean of 0 and a standard deviation of 1.***

Examples cases of Using Machine Learning

- ***Machine learning is a powerful tool that can be used in a wide range of applications such as:***

- ***Predictive modeling:*** Machine learning can be used to build predictive models that can predict future outcomes based on historical data. This can be used in many applications, such as ***stock market prediction***, ***fraud detection***, ***weather forecasting***, ***and customer behavior prediction***.

- ***Image recognition:*** Machine learning can be used to train models that can recognize ***objects, faces, and other patterns in images***. This is used in many applications, such as ***self-driving cars, facial recognition systems, and medical image analysis***.

Examples cases of Using Machine Learning

- ***Natural language processing:*** Machine learning can be used to analyze and understand natural language, which is used in many applications, such as ***chatbots, voice assistants, and sentiment analysis***.

- ***Recommendation systems:*** Machine learning can be used to build recommendation systems that can suggest ***products, services, or content to users based on their past behavior or preferences.***

- ***Data analysis:*** Machine learning can be used to analyze large datasets and identify patterns and insights that would be ***difficult or impossible for humans to detect.***

- ***Robotics:*** Machine learning can be used to train robots to perform tasks autonomously, such as ***navigating through a space or manipulating objects.***

Examples cases of Using Machine Learning

- ***Self-Driving Cars:*** These rely on reinforcement learning, a type of machine learning where algorithms learn through trial and error in a simulated environment.

- ***Medical Diagnosis:*** Machine learning algorithms can analyze medical images (X-rays, MRIs) to identify abnormalities and aid doctors in diagnosis.

- ***Benefits and Challenges of Machine Learning:***

Machine learning (ML) has become a transformative technology across various industries. While it offers numerous advantages, it's crucial to acknowledge the challenges that come with its increasing use.

Benefits of Machine Learning

- ***Enhanced Efficiency and Automation:*** Automates repetitive tasks, freeing up human resources for more complex work. It also streamlines processes, leading to increased efficiency and productivity.

- ***Data-Driven Insights:*** Can analyze vast amounts of data to identify patterns and trends that humans might miss. This allows for better decision-making based on real-world data.

- ***Improved Personalization:*** Personalizes user experiences across various platforms. From recommendation systems to targeted advertising, ML tailors' content and services to individual preferences.

- ***Advanced Automation and Robotics:*** Empowers robots and machines to perform complex tasks with greater accuracy and adaptability. This is revolutionizing fields like manufacturing and logistics.

Challenges of Machine Learning

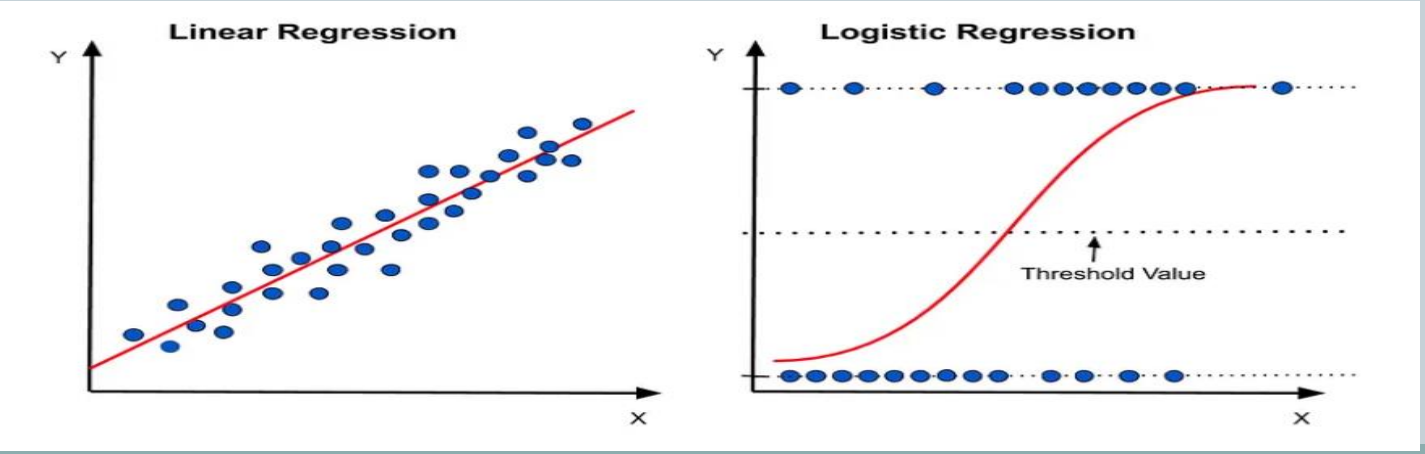
- ***Data Bias and Fairness:*** Algorithms are only as good as the data they are trained on. Biased data can lead to discriminatory outcomes, requiring careful data selection and monitoring of algorithms.
- ***Security and Privacy Concerns:*** Relies heavily on data, security breaches can expose sensitive information. Additionally, the use of personal data raises privacy concerns that need to be addressed.
- ***Interpretability and Explainability:*** Complex ML models can be difficult to understand, making it challenging to explain their decision-making processes. This lack of transparency can raise questions about accountability and trust.
- ***Job Displacement and Automation:*** Automation through ML can lead to job displacement in certain sectors. Addressing the need for retraining and reskilling the workforce is crucial.

Examples of Supervised Learning Algorithms

- ***Supervised learning algorithms divided into several different types, each with its own unique characteristics and applications***:

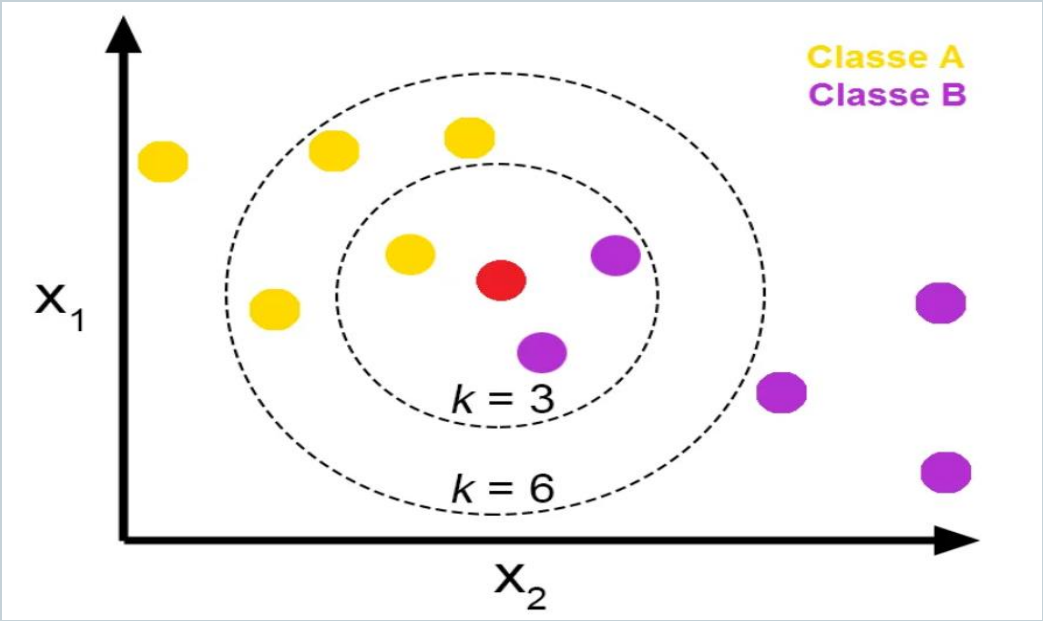
- ***Linear Regression:*** Linear regression is a type of supervised learning regression algorithm that is used to predict a continuous output value. It is one of the simplest and most widely used algorithms in supervised learning.

- ***Logistic Regression :*** Logistic regression is a type of supervised learning classification algorithm that is used to predict a binary output variable.



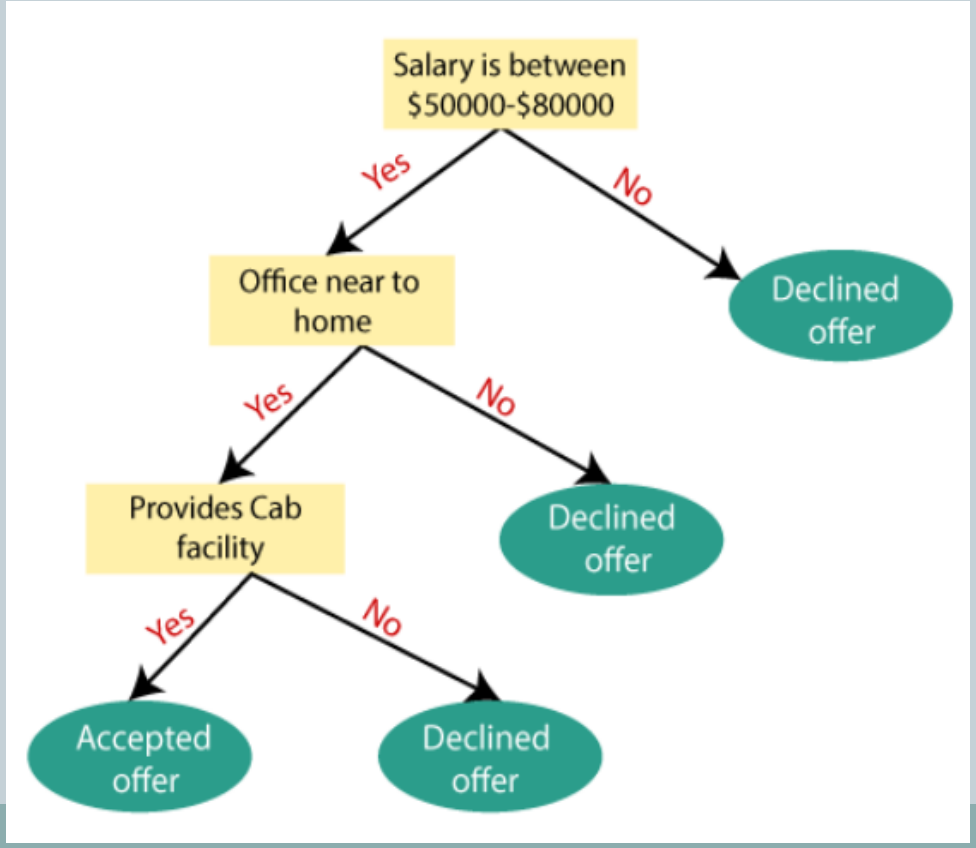
Examples of Supervised Learning Algorithms

- ***K-Nearest Neighbors (KNN) :*** KNN works by finding k training examples closest to a given input and then predicts the class or value based on the majority class or average value of these neighbors. The performance of KNN can be influenced by the choice of k and the distance metric used to measure proximity.



Examples of Supervised Learning Algorithms

- ***Decision Trees :*** Decision tree is a tree-like structure that is used to model decisions and their possible consequences. Each internal node in the tree represents a decision, while each leaf node represents a possible outcome.

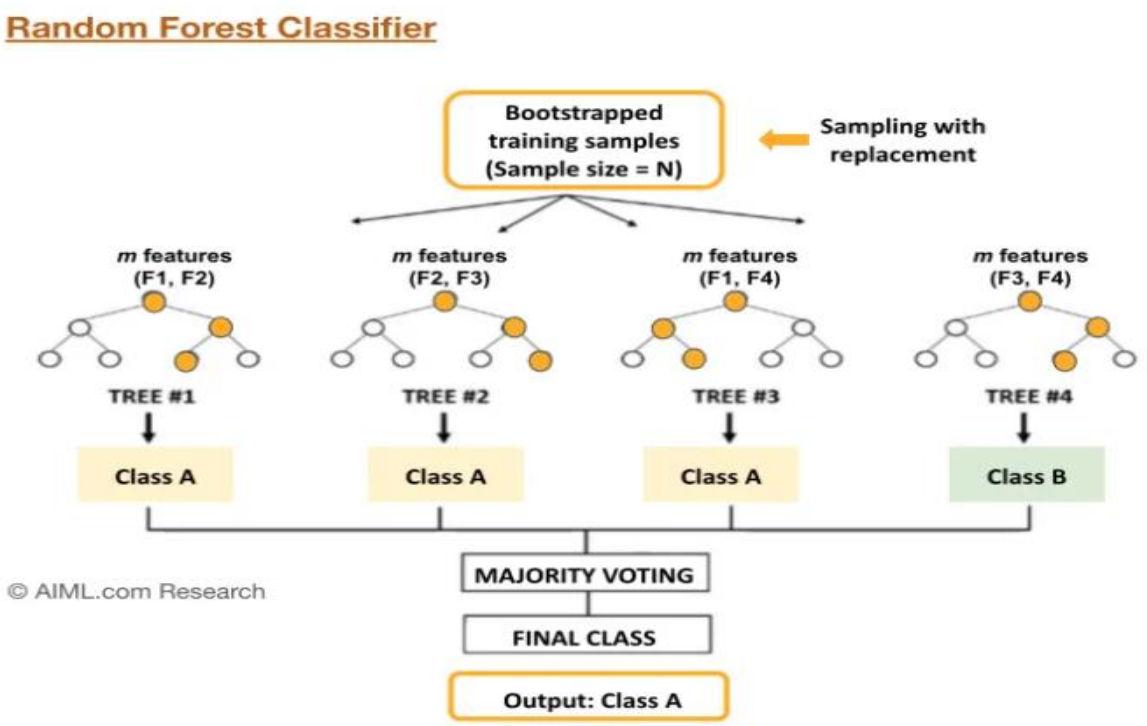


Examples of Supervised Learning Algorithms

- ***Random Forests :*** Random forests again are made up of multiple decision trees that work together to make predictions. Each tree in the forest is trained on a different subset of the input features and data. The final prediction is made by aggregating the predictions of all the trees in the forest.

Training Data
(Sample size, N= 6, No. of features, F = 4)

F1	F2	F3	F4	Y
2.1	0	400	-9	A
3.0	1	890	-42	B
2.2	1	929	0	B
4.0	0	324	-23	A
3.5	1	333	-15	A
6.0	0	215	-9	A

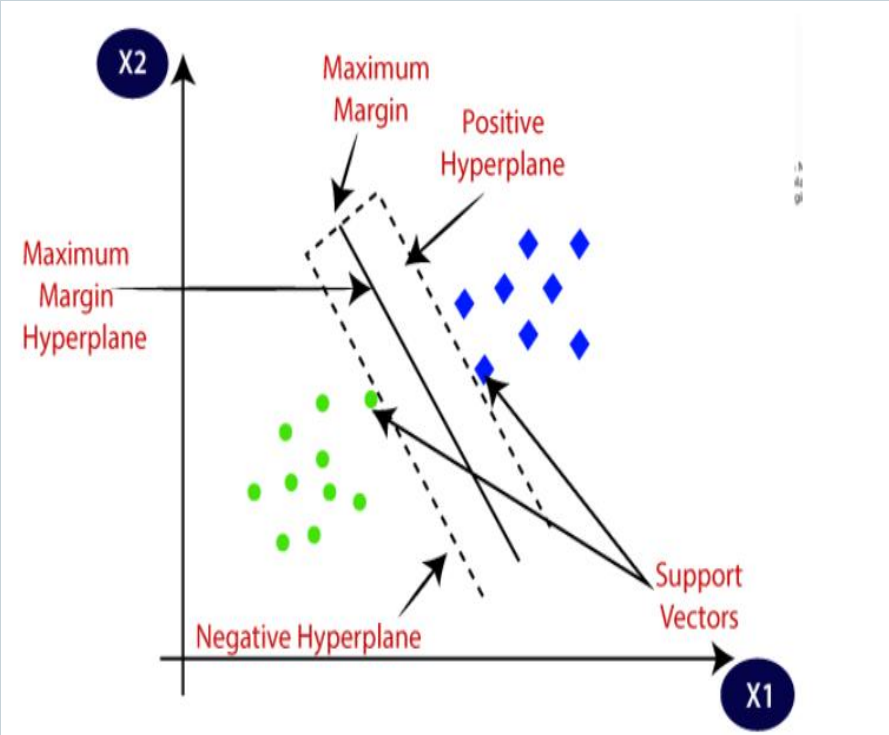


Key parameters of Random Forest Model are: (a) Number of trees , (b) Maximum depth of the trees (c) Size of the random subset of features
In this example, No. of trees = 4, Depth = 2, and Feature subset size, m = 2 (no. of features/2)

Examples of Supervised Learning Algorithms

- ***Support Vector Machine(SVM) :*** The SVM algorithm creates a hyperplane to segregate n-dimensional space into classes and identify the correct category of new data points. The extreme cases that help create the hyperplane are called support vectors, hence the name Support Vector Machine.

- ***Support Vector Machines (SVMs) aim to find a decision boundary (hyperplane) that maximally separates data points belonging to different classes. In Support Vector Machines (SVMs), the main idea is the margin. The margin signifies the distance between the decision boundary and the closest data points from each class***



Examples of Supervised Learning Algorithms

- ***Naive Bayes Algorithm:*** The Naive Bayes algorithm is a supervised machine learning algorithm based on applying Bayes' Theorem with the "naive" assumption that features are independent of each other given the class label.

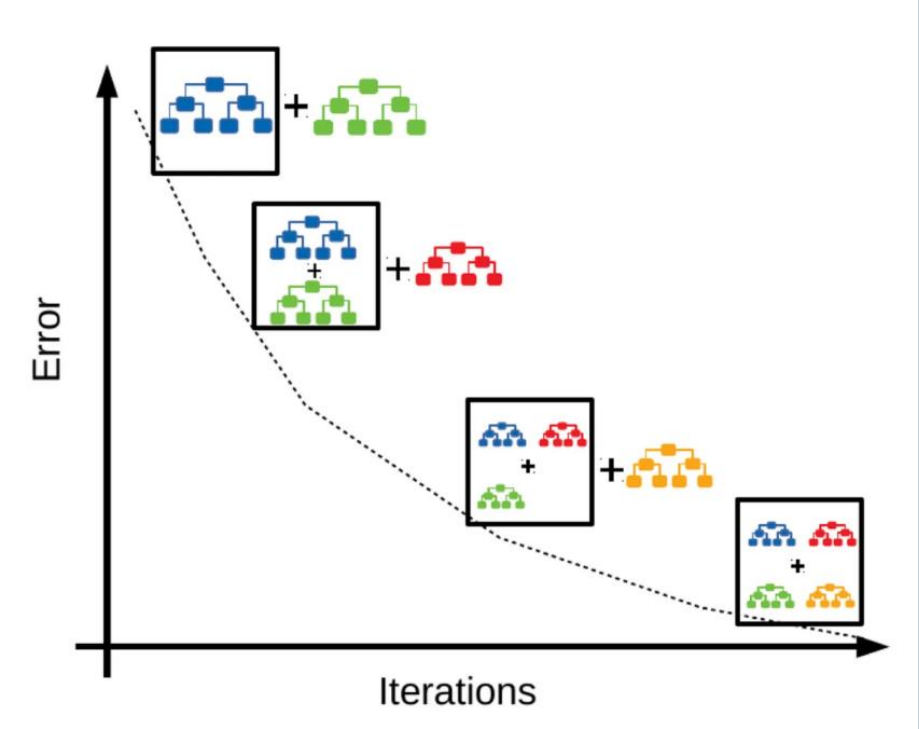
According to Bayes, this conditional probability can be calculated using the following formula:

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

- P(B|A) = probability that event B occurs if event A has already occurred
- P(A) = probability that event A occurs
- P(B) = probability that event B occurs

Examples of Supervised Learning Algorithms

- ***Gradient Boosting :*** It combines weak learners, like decision trees, to create a strong model. It iteratively builds new models that correct errors made by previous ones.



Understanding Linear Regression

- ***In simple terms, Linear Regression exploited the power of straight lines.***

- Imagine you are a realtor trying to predict the price of a house. You might consider various factors like its ***size, location, age and number of bedrooms.***

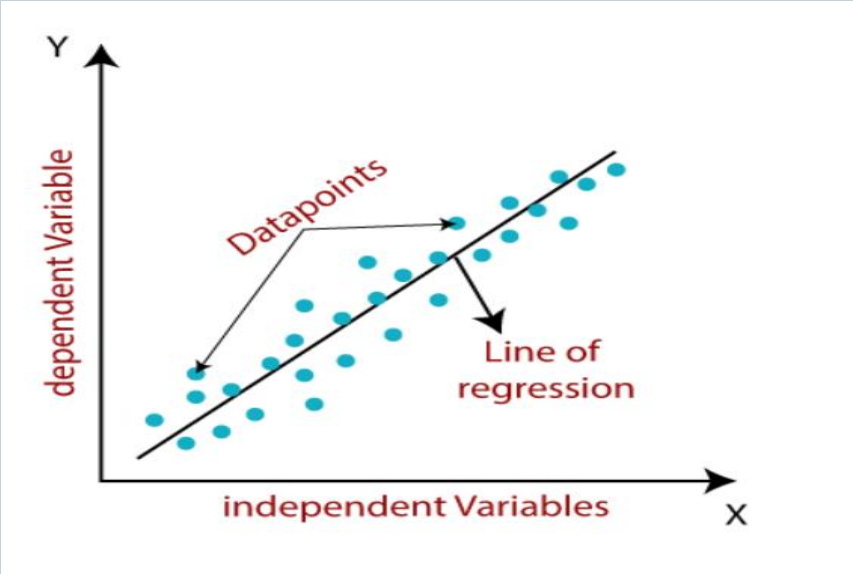
- Linear regression comes in as a powerful tool to analyze these factors and detect the ***underlying relationships***. It is a powerful statistical technique that allows us to examine the relationship between one or more "***independent***" variables and a "***dependen***t" variable.

- In a ***house price dataset***, the independent variables are columns used to predict, such as the "Area", "Bedrooms", "Age", and "Location". The Dependent variable will be the "Price" column – the feature to be predicted.

Understanding Linear Regression

- ***Linear regression*** is the simplest form of regression, assuming a linear (straight line) relationship between the input and the output variable. The equation for simple linear regression can be expressed as ***y = mx + b***, where, ***y*** is the dependent variable, ***x*** is the independent variable, ***m*** is the ***slope***, and ***b*** is the ***intercept***. This creates a ***best fit line***,

  - ***y is the dependent variable***
  - ***x is the independent variable***
  - ***m is the slope***
  - ***and b is the intercept.***



Why Linear Regression is Important?

- ***Interpretability:*** Unlike some complex models, linear regression provides clear insights into how each feature influences the target variable. You can easily see which features have the strongest impact and how they contribute to the overall prediction.

- ***Baseline for complex models:*** When dealing with more intricate problems, data scientists often start with linear regression as a baseline model. This simple model serves as a reference point to compare the performance of more complex algorithms. If a complex model doesn't offer significantly better results than linear regression, it might be unnecessarily overfitting the data.

- ***Ease of implementation:*** Compared to other machine learning algorithms, linear regression is relatively easy to implement and understand. This makes it a great starting point for beginners moving into the world of machine learning.

What is the best Fit Line?

- ***Hint:*** linear regression might not be the ultimate solution for every problem, but it offers a powerful foundation for understanding data, building predictions, and setting the stage for exploring more complex models.

- ***Note:*** The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).

Linear Regression Key Concepts

To create a Linear Regression Model:

- ***Find the best fit line:*** Lines are drawn across a graph, with features on one axis and prices on the other. The line we're looking for is the one that best fits the dots representing real houses, minimizing the overall difference between predicted and actual prices.

- ***Minimize the error:*** Think of the line as a balancing act. The line's slope and position is adjusted until the total distance between the line and the data points is as small as possible (***Minimized Cost Function***). This minimized distance reflects the best possible prediction for new houses based on their features.

Linear Regression Key Concepts

- ***Coefficients:*** Each feature in the model gets a weight (Coefficients), like a specific amount of an ingredient in a recipe. By adjusting these weights, we change how much each feature contributes to the predicted price. A higher weight for size, for example, means that larger houses tend to have a stronger influence on the predicted price.

- ***Note:*** The ***loss function*** is to capture the difference between the actual and predicted values for a single record whereas ***cost functions*** aggregate the difference for the entire training dataset.

Feature weights

- ***Once we have the best-fit line, the model can predict the price of new houses based on their features.***

- The ***weights of features*** is important too, They reveal how much, on average, each feature changes the predicted price. ***A positive weight for bedrooms means that, generally, houses with more bedrooms are predicted to be more expensive.***

Assumptions and Limitations

- ***Linear regression*** assumes things are roughly straightforward, like the relationship between size and price. If things are more complex, it might not be the best tool.
- But it's a great starting point because it's easy to understand and interpret, making it a valuable tool to explore the world of data prediction.
- However, you do not have to worry about finding the best fit line manually. The algorithm picks the best fit line when creating the model.

Prerequisites for Implementing LR

- ***Before we begin, make sure you have the following installed:***
- ***Python*** (3.x recommended)
- ***Jupyter Notebook :*** this is optional but recommended for an interactive environment and also for trial and error (beginners will benefit the most from this)
- ***Required libraries:*** pandas, NumPy, Matplotlib, seaborn, scikit-learn

Prerequisites for Implementing LR

- ***You can install these libraries using pip install in the command line:***

```
pip install pandas
pip install matplotlib
pip install numpy
pip installseaborn
pip install scikit-learn
```

How to Build Your First Model

- **How to import libraries and load data**
- ***The libraries are imported as abbreviations for the sole purpose of writing shorter code:***

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load your dataset (replace 'your_dataset.csv' with your file)
df = pd.read_csv('your_dataset.csv')

# Display the first few rows of the dataset
df.head()
```

- The dataset is loaded using pandas' read_csv function and then the first five rows are displayed using df.head().

How to Build Your First Model

- ***Check for missing values***
- ***Machine learning models cannot function when there are missing values in the dataset:***

```
df.isnull().sum()
```

- ***This will give you a list of columns that have null values and the rows themselves. There are different ways to deal with this such as:***

- Deleting all rows with null values.
- Using the mean or median of the column to fill in the missing values for numerical data.
- Filling the missing values with the most occurring data for qualitative data.

How to Build Your First Model

- ***Explore the correlation between variables***

```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```

- ***This code will show the relationships between the columns of independent / variables / features, and dependent/ target variables.***

- ***It will also show which columns or features determine the outcome of the target variable more than others.***

How to Build Your First Model

❑ Explore the correlation between variables

```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```

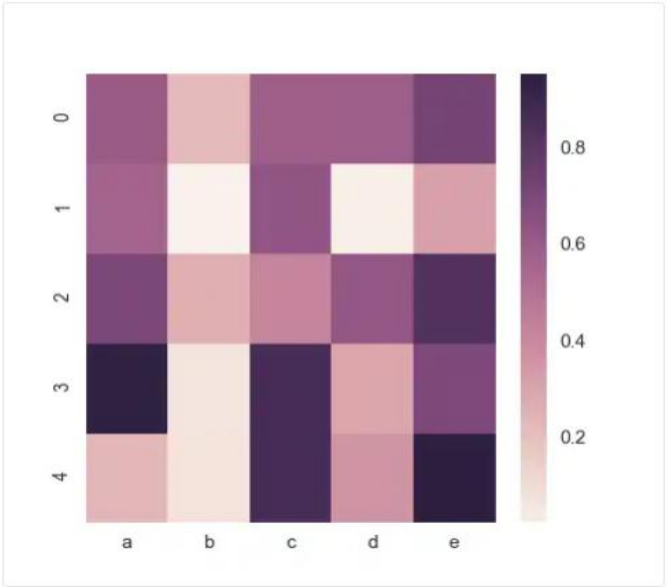
- A ***heatmap*** is a graphical representation of data where each value of a matrix is represented as a color.
- Python ***Annotations (annot)*** are a Python feature that tells developers the data type of variables or function parameters. They can also be used to improve the readability of a Python program. There are two main types of Python annotations: function annotations and variable annotations.
- A ***colormap (cmap)*** is a matrix of colors used to represent data values in a plot. They can be used in various types of plots, including maps, heatmaps, and scatter plots, to help interpret data more easily

Example of code for using the heatmap() function of Seaborn

```
# library
import seaborn as sns
import pandas as pd
import numpy as np

# Create a dataset
df = pd.DataFrame(np.random.random((5,5)), columns=["a","b","c","d","e"])

# Default heatmap
p1 = sns.heatmap(df)
```



- ***np.random.random((5,5)):*** Generates a 5×5 NumPy array with random values between 0 and 1.
- ***pd.DataFrame***(..., columns=["a","b","c","d","e"]): Converts the NumPy array into a Pandas DataFrame, naming the columns "a" to "e".
- ***sns.heatmap(df):*** Uses Seaborn's heatmap function to visualize the data as a heatmap.Darker or lighter colors represent lower or higher values, respectively.

Data Visualization

- ***Visualize the relationship between independent and dependent variables***

  - Scatter plots can show how well your predicted prices align with actual values. Residual plots help visualize any patterns in the errors, revealing potential issues.

```
sns.scatterplot(x='Independent Variable', y='Dependent Variable', data=df)
plt.show()
```

  - This scatter plot shows the relationship between independent and dependent variables and a straight line is drawn to show the relationship

Data preprocessing

- This is a crucial step as the quality of data that is used to train the model also determines the accuracy and efficiency of the model.
- Here, the data set is ***first separated into X*** (independent variable(s)/ features) and ***Y label*** (dependent variable/ Target).
- We handle the missing values by dropping columns with missing/ null values and split the dataset into training and testing in an 80:20 ratio.

```
# Handle missing values if necessary
df.dropna(inplace=True)

# Split the data into features (X) and target variable (y)
X = df[['Independent Variable']]
y = df['Dependent Variable']

# Split the data into training and testing sets (e.g., 80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Building the Regression Model

- We create a model by calling an instance of the model into a variable as shown below and train the model by fitting the training dataset into the model.

```
# Create a linear regression model
model = LinearRegression()

# Train the model
model.fit(X_train, y_train)
```

How to make Predictions?

- The trained model is used to make predictions on the test set. Predictions can be made on the entire feature column as shown below or each column can be predicted individually.

```
# Make predictions
y_predn = model.predict(X_test)
```

How to evaluate the Model

- ***Evaluating the model's performance is an important step to determine the accuracy of the model and reusability. We can check using metrics such as:***

- "***R-squared***": This tells you how well the model explains the variation in house prices. A higher value (closer to 1) indicates a better fit.

- ***Mean Squared Error (MSE):*** This measures the average difference between predicted and actual prices. Lower is better.

- ***Precision score*** measures the accuracy of positive predictions. It answers the question, "Of all the items the model labeled as positive, how many were actually positive?"

```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Using precision score
model.score(X_test, Y_test)
```

How to Visualize the Results

- ***Visualize the regression line and actual vs. predicted values:***

```
# Plot the regression line
plt.scatter(X_test, y_test, color='gray')
plt.plot(X_test, y_pred, color='red', linewidth=2)
plt.xlabel('Independent Variable')
plt.ylabel('Dependent Variable')
plt.show()
```

H.W: ***Consider a different scenario where you want predict a student's score (y) based on the number of hours they studied (x).***