## THE POWER OF VISUALIZATION IN MACHINE LEARNING!





SHIVAM MODI
@learneverythingai



Step into the world of linear regression and discover how visualizing data points and regression lines help us understand the relationships between variables. From the classic "y = mx + b" formula to practical Python code, let's uncover the art of turning data into meaningful predictions!

from sklearn.linear\_model import LinearRegression

# Assuming X and y are your feature and target variables
regressor = LinearRegression()
regressor.fit(X, y)





Discovering the Power of Scatter Plots

In this slide, we delve into the elegance of scatter plots, where data points become stars in the sky of visualization. We'll unveil the simplicity of the "Plot(x, y)" formula and demonstrate how to create captivating scatter plots using Python. Get ready to connect the dots and reveal exciting patterns!

```
import matplotlib.pyplot as plt
```

```
# Assuming data_x and data_y are your data points
plt.scatter(data_x, data_y)
plt.xlabel('X-axis label')
plt.ylabel('Y-axis label')
plt.title('Scatter Plot Example')
plt.show()
```





Unleashing the Potential of Histograms

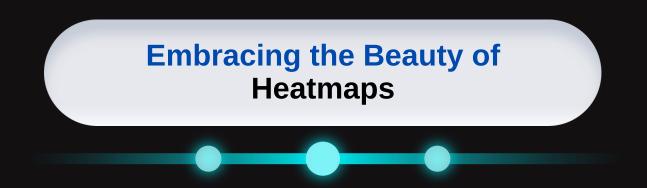
Journey into the realm of histograms, where data distributions take center stage. From the straightforward "Hist(data, bins)" formula to a practical Python snippet, we'll learn how to visualize the frequency of occurrences and gain valuable insights into our data!

#### import matplotlib.pyplot as plt

```
# Assuming data is your dataset and bins is the number of bins
plt.hist(data, bins=10)
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram Example')
plt.show()
```







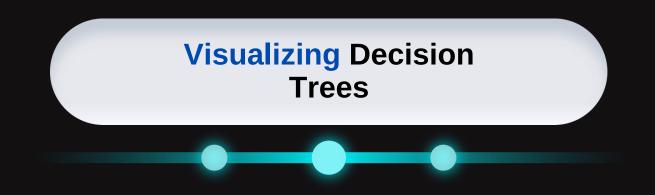
In this visually stunning slide, we'll uncover the magic of heatmaps – a powerful way to represent correlation and multidimensional data. Embrace the "Heatmap(data)" formula and follow the Python code to transform complex information into a work of art!

import seaborn as sns
import matplotlib.pyplot as plt

# Assuming data is your correlation matrix sns.heatmap(data, annot=True, cmap='coolwarm') plt.title('Correlation Heatmap') plt.show()







Step into the world of decision trees and witness how these versatile models come to life through visualization. From the formula "DecisionTreeClassifier()" to Python code that showcases their branching logic, let's decode the secrets of decision-making trees!





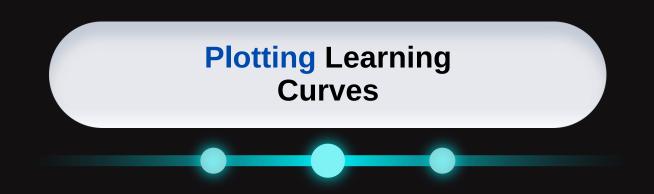


Explore the captivating world of K-Means clustering, where data points find their tribes. Unravel the formula "KMeans(n\_clusters=k)" and follow the Python script to map clusters on a colorful canvas. Join us on this geographical journey through data!

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
# Assuming X is your data and k is the number of clusters
kmeans = KMeans(n_clusters=k)
kmeans.fit(X)
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_, cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200, c='black',
label='Centroids')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('K-Means Clustering')
plt.legend()
plt.show()
```







In this slide, we delve into the learning curve, a valuable tool to assess a model's performance and diagnose underfitting or overfitting. Experience the formula "LearningCurve(model)" and master Python code that illustrates the evolution of accuracy as knowledge expands!

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
def plot_learning_curve(model, X, y):
   train_sizes, train_scores, test_scores = learning_curve(model, X, y, train_sizes=np.linspace(0.1,
1.0, 5))
    train_mean = np.mean(train_scores, axis=1)
    train std = np.std(train scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)
    test_std = np.std(test_scores, axis=1)
    plt.plot(train_sizes, train_mean, color='blue', marker='o', markersize=5, label='Training
Accuracy')
    plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std, alpha=0.15,
color='blue')
    plt.plot(train_sizes, test_mean, color='green', linestyle='--', marker='s', markersize=5,
label='Validation Accuracy')
    plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std, alpha=0.15,
color='green')
    plt.xlabel('Training Samples')
    plt.ylabel('Accuracy')
    plt.title('Learning Curve')
    plt.legend()
    plt.grid(True)
    plt.show()
```



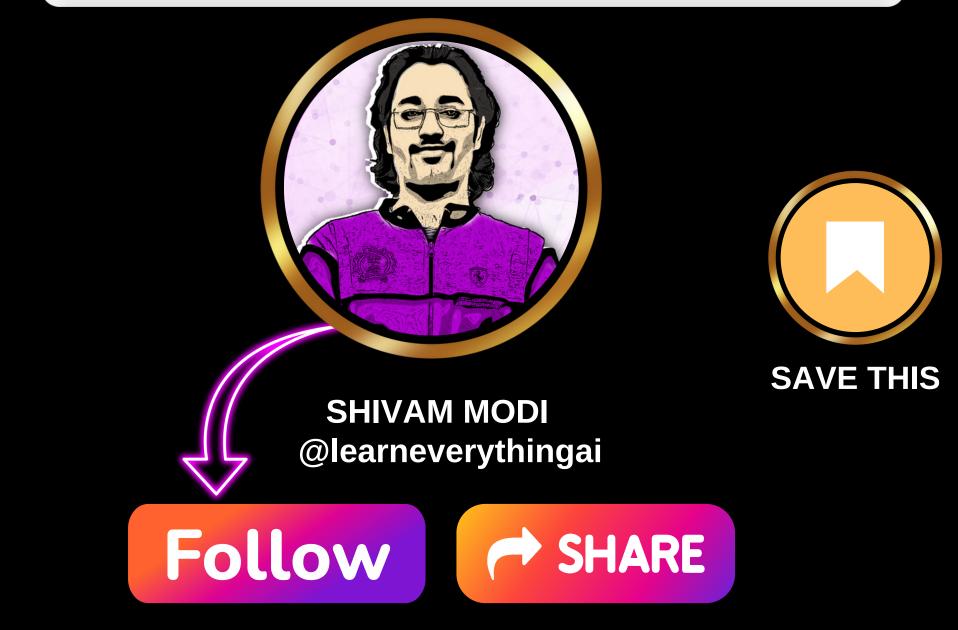




#### @learneverythingai

# Like this Post?

- Follow Me
- Share with your friends
- Check out my previous posts



www.learneverythingai.com